I.J. Image, Graphics and Signal Processing, 2013, 5, 55-62
Published Online April 2013 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijigsp.2013.05.07

A Comparative Analysis of Image Scaling Algorithms

Chetan Suresh
Department of Electrical and Electronics Engineering, BITS Pilani
Pilani - 333031, Rajasthan, India
E-mail: shivchetan@gmail.com

Sanjay Singh, Ravi Saini, Anil K Saini
Scientist, IC Design Group,
CSIR – Central Electronics Engineering Research Institute (CSIR-CEERI)
Pilani – 333031, Rajasthan, India

***Abstract*—** Image scaling, fundamental task of numerous image processing and computer vision applications, is the process of resizing an image by pixel interpolation. Image scaling leads to a number of undesirable image artifacts such as aliasing, blurring and moiré. However, with an increase in the number of pixels considered for interpolation, the image quality improves. This poses a quality-time trade off in which high quality output must often be compromised in the interest of computation complexity. This paper presents a comprehensive study and comparison of different image scaling algorithms. The performance of the scaling algorithms has been reviewed on the basis of number of computations involved and image quality. The search table modification to the bicubic image scaling algorithm greatly reduces the computational load by avoiding massive cubic and floating point operations without significantly losing image quality.

***Index Terms*—** Image Scaling, Nearest-neighbour, Bilinear, Bicubic, Lanczos, Modified Bicubic

I. INTRODUCTION

Image scaling is a geometric transformation used to resize digital images and finds widespread use in computer graphics, medical image processing, military surveillance, and quality control [1]. It plays a key role in many applications [2] including pyramid construction [3]-[4], super-sampling, multi-grid solutions [5], and geometric normalization [6].

In surveillance-based applications, images have to be monitored at a high frame rate. Since, the images need not be of the same size, image scaling is necessary for comparison and manipulation of images. However, image scaling is a computationally intensive process due to the convolution operation, which is necessary to band-limit the discrete input and thereby diminishes undesirable aliasing artifacts [2].

Various image scaling algorithms are available in literature and employ different interpolation techniques to the same input image. Some of the common interpolation algorithms are the nearest neighbour, bilinear [7], and bicubic [8]-[9]. Lanczos algorithm utilizes the 3-lobed Lanczos window function to implement interpolation [10].

There are many other higher order interpolators which take more surrounding pixels into consideration, and thus also require more computations. These algorithms include spline [11] and sinc interpolation [12], and retain the most of image details after an interpolation. They are extremely useful when the image requires multiple rotations/distortions in separate steps. However, for single-step enlargements or rotations, these higher-order algorithms provide diminishing visual improvement and processing time increases significantly.

Novel interpolation algorithms have also been proposed such as auto-regression based method [13], fuzzy area-based scaling [14], interpolation using classification and stitching [15], isophote-based interpolation [16], and even interpolation scheme combined with Artificial Neural Networks [17]. Although these algorithms perform well, they require a lengthy processing time due to their complexity. This is intolerable for real-time image scaling in video surveillance system. Hence, these algorithms have not been considered for the comparative analysis in this paper.

In this paper, firstly, image interpolation algorithms are classified and reviewed; then evaluation and comparison of five image interpolation algorithms are discussed in depth based on the reason that evaluation of image interpolation is essential in the aspect of designing a real-time video surveillance system. Analysis results of the five interpolation algorithms are summarized and presented.

II. IMAGE SCALING

Image scaling is obtained by performing interpolation over one or two directions to approximate a pixel's colour and intensity based on the values at neighbouring pixels. More the adjacent pixels used for interpolation, better the quality of interpolated output image.

Digital image interpolation is the process of generating a continuous intensity surface from discrete image data samples. Generally, almost every geometric transformation like translating, rotating, scaling, and warping requires interpolation to be performed on an image [18]. Such transformations are essential to any commercial digital image processing software. The perceived quality of the interpolated images is affected by several issues such as sharpness of edges, freedom from artifacts and reconstruction of high frequency details [18].

Interpolation algorithms attempt to generate continuous data from a set of discrete data samples through an interpolation function. These interpolation methods aim to minimize the visual defects arising from the inevitable resampling error and improve the quality of re-sampled images [19]. Interpolation function is performed by convolution operation which involves a large number of addition and multiplication operations. Hence, a trade-off is required between computation complexity and quality of the scaled image.

Based on the content awareness of the algorithm, image scaling algorithms can be classified as Adaptive image scaling and Non-adaptive image scaling.

Adaptive image scaling algorithms modify their interpolation technique based on the image content being a smooth texture [20] or a sharp edge [21]-[22]. As the interpolation method changes in real-time, these algorithms are complex and computationally intensive. They find widespread use in image editing software as they ensure a high quality scaled image. As the focus is on analysing image scaling algorithms for real-time applications, these algorithms are beyond the scope of the paper.

Non-adaptive image scaling algorithms like nearest neighbour, bilinear, bicubic and Lanczos algorithms have a fixed interpolation method irrespective of the image content.

III. Comparative Analysis of Image Scaling Algorithms

This section discusses the Non-adaptive Image scaling algorithms like nearest neighbour, bilinear, bicubic, modified bicubic, and Lanczos. Let P(x,y) represent the pixel in input image of MxN dimensions and U(i,j) represents the pixel in scaled/resized output image of size M'xN'. Let T be the transformation used for scaling. The each pixel of output image is computed by applying the transformation T on input image pixels as shown in Fig. 1.

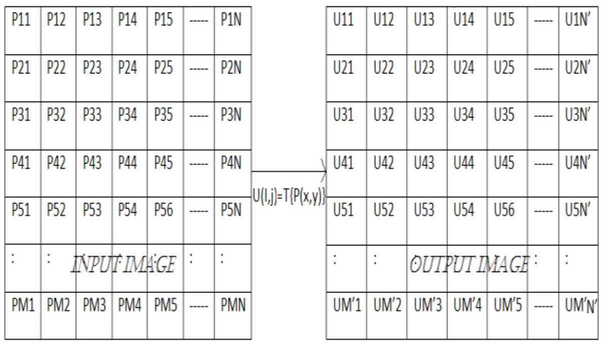
P11 P12 P13 P14 P15 ---- P1N
P21 P22 P23 P24 P25 ---- P2N
P31 P32 P33 P34 P35 ---- P3N
P41 P42 P43 P44 P45 ---- P4N
P51 P52 P53 P54 P56 ---- P5N
INPUT IMAGE
PM1 PM2 PM3 PM4 PM5 ---- PMN
U(i,j)=T(P(x,y))
U11 U12 U13 U14 U15 ---- U1N'
U21 U22 U23 U24 U25 ---- U2N'
U31 U32 U33 U34 U35 ---- U3N'
U41 U42 U43 U44 U45 ---- U4N'
U51 U52 U53 U54 U56 ---- U5N'
OUTPUT IMAGE
UM'1 UM'2 UM'3 UM'4 UM'5 ---- UM'N'

Figure 1. Image scaling procedure

A. Nearest Neighbour Image Scaling

The simplest image interpolation method is to select the value of the nearest input image pixel and assigning the same value to scaled image pixel. The nearest neighbour algorithm (Fig. 2) selects the value of the nearest known pixel and does not consider the values of other neighbouring pixels, yielding a piecewise constant function [23].

For nearest neighbour algorithm the level of complexity is low leading to a fast computation time. But the scaled image quality is low and high aliasing effect is observed.

```
float tx = width_source/width_dst;
float ty = height_source/ height_dst;

for(i=0; i<height_dst; i++)
 for(j=0; j<width_dst; j++)
  {
      x = ceil(j*tx);
      y = ceil(i*ty);
      U(i,j) = P(y,x);
  }
```

Figure 2. Nearest neighbour image scaling algorithm

B. Bilinear Image Scaling

Another simple interpolation method is linear interpolation, a slight improvement over the nearest neighbour interpolation algorithm. In linear interpolation, the interpolated value is computed based on the weighted average of two data points. The weights are inversely proportional to the distance of the end points from the interpolated data point.

Bilinear interpolation algorithm [7] performs linear interpolation in one direction followed by linear interpolation of the interpolated values in the other direction. The algorithm uses 4 diagonal pixels surrounding the corresponding source image pixel for interpolation (Fig. 3). It computes a weighted average depending on the nearness and brightness of the 4 diagonal pixels and assigns that value to the pixel in the output image.

```
float tx = (width_source)/width_dst;
float ty = (height_source)/ height_dst;
float x_diff, y_diff;

for(i=0; i<height_dst; i++)
 for(j=0; j<width_dst; j++)
  {
    x = (int)(tx * j);
    y = (int)(ty * i);

    x_diff = ((tx * j) -x);
    y_diff = ((ty * i) -y );

    U(i,j) = P(y,x) *(1-x_diff)*(1-y_diff) +
             P(y,x+1)*(1-y_diff)*(x_diff) +
             P(y+1,x)*(y_diff)*(1-x_diff) +
             P(y+1,x+1)*(y_diff)*(x_diff);
 }
```

Figure 3. Bilinear image scaling algorithm

Bilinear offers considerable improvement in image quality with a slightly increased complexity. The aliasing effect is reduced though blurring is observed.

C. Bicubic Image Scaling

Bicubic interpolation algorithm [8]-[9] performs cubic interpolation in one direction followed by cubic interpolation of the interpolated values in the other direction (Fig. 4).

```
float tx = (width_source)/width_dst;
float ty = (height_source)/ height_dst;

for(i=0; i<height_dst; i++)
 for(j=0; j<width_dst; j++)
  {
      x =(int)(tx*j);
      y =(int)(ty*i);

      dx=(float)(tx*j-x);
      dy=(float)(ty*i-y);

      for(jj=0;jj<=3;jj++)
        {
          d0 = P(y-1+jj,x-1) - P(y-1+jj,x);
          d2 = P(y-1+jj,x+1) - P(y-1+jj,x);
          d3 = P(y-1+jj,x+2) - P(y-1+jj,x);
          a0 = P(y-1+jj,x);
          a1 = (-1.0/3*d0 + d2 -1.0/6*d3);
          a2 = (1.0/2*d0 + 1.0/2*d2);
          a3 = (-1.0/6*d0 - 1.0/2*d2 + 1.0/6*d3);
          C[jj]=(a0 + a1*dx + a2*dx*dx + a3*dx*dx*dx);
        }
          d0 = (C[0]-C[1]);
          d2 = (C[2]-C[1]);
          d3 = (C[3]-C[1]);
          a0 = C[1];
          a1 = ( -1.0/3*d0 + d2 -1.0/6*d3);
          a2 = (1.0/2*d0 + 1.0/2*d2);
          a3 = (-1.0/6*d0 - 1.0/2*d2 + 1.0/6*d3);
          Cc = (a0 + a1*dy + a2*dy*dy + a3*dy*dy*dy);
          if(Cc>255) Cc=255;
          if(Cc<0) Cc=0;
          U(i,j) = Cc
   }
```

Figure 4. Bicubic image scaling algorithm

The bicubic interpolation approximates a sinc interpolation by using cubic polynomial waveforms instead of linear waveforms when computing an output pixel value. The algorithm uses 16 nearest pixels surrounding the closest corresponding pixel in the source image for interpolation.

Bicubic offers high scaled image quality at the cost of considerably high complexity. Edge halo effect is observed though aliasing and blurring are reduced.

D. Lanczos Image Scaling

The Lanczos interpolation algorithm [10] uses a windowed form of sinc filter (an ideal low-pass filter) to perform interpolation on a 2-D grid. Sinc function of a variable can be mathematically obtained by dividing the sine function of the variable by itself [9]. Sinc function theoretically never goes to zero. Hence, a practical filter can be implemented by multiplying the sinc function with a window operator such as Hamming [24]-[25] and Hann [26] to obtain a filter with a finite size.

The Lanczos filter can be obtained by multiplying the sinc function with the Lanczos window which is truncated to zero outside of the main lobe [27]. The size of the Lanczos window is defined by the order of the convolution kernel.

```
float tx = (width_source)/width_dst;
float ty = (height_source)/ height_dst;

for(i=0; i<height_dst; i++)
 for(j=0; j<width_dst; j++)
  {
    x =(int)(tx*j);     y =(int)(ty*i);
    dx = tx*j - x;      dy = ty*i - y;

    a0 = sin(pi*(dx+2))*sin(pi*(dx+2)/3)/(pi*pi*(dx+2)*(dx+2)/3);
    a1 = sin(pi*(dx+1))*sin(pi*(dx+1)/3)/(pi*pi*(dx+1)*(dx+1)/3);
    a2 = sin(pi*(dx))*sin(pi*(dx)/3)/(pi*pi*(dx)*(dx)/3);
    a3 = sin(pi*(dx-1))*sin(pi*(dx-1)/3)/(pi*pi*(dx-1)*(dx-1)/3);
    a4 = sin(pi*(dx-2))*sin(pi*(dx-2)/3)/(pi*pi*(dx-2)*(dx-2)/3);
    a5 = sin(pi*(dx-3))*sin(pi*(dx-3)/3)/(pi*pi*(dx-3)*(dx-3)/3);
    b0 = sin(pi*(dy+2))*sin(pi*(dy+2)/3)/(pi*pi*(dy+2)*(dy+2)/3);
    b1 = sin(pi*(dy+1))*sin(pi*(dy+1)/3)/(pi*pi*(dy+1)*(dy+1)/3);
    b2 = sin(pi*(dy))*sin(pi*(dy)/3)/(pi*pi*(dy)*(dy)/3);
    b3 = sin(pi*(dy-1))*sin(pi*(dy-1)/3)/(pi*pi*(dy-1)*(dy-1)/3);
    b4 = sin(pi*(dy-2))*sin(pi*(dy-2)/3)/(pi*pi*(dy-2)*(dy-2)/3);
    b5 = sin(pi*(dy-3))*sin(pi*(dy-3)/3)/(pi*pi*(dy-3)*(dy-3)/3);

    for(jj=0;jj<=5;jj++)
     {
       d0 = P(y-2+jj,x-2);      d1 = P(y-2+jj,x-1);
       d2 = P(y-2+jj,x);        d3 = P(y-2+jj,x+1);
       d4 = P(y-2+jj,x+2);      d5 = P(y-2+jj,x+3);
       C[jj]= a0*d0 + a1*d1 + a2*d2 + a3*d3 + a4*d4 +a5*d5;
     }

     Cc = a0*C[0]+a1*C[1]+a2*C[2] +a3*C[3]+a4*C[4]+a5*C[5];
     if(Cc>255) Cc=255;    if(Cc<0) Cc=0;
     U(i,j) = Cc;
}
```

Figure 5. Lanczos image scaling algorithm

The number of neighbouring pixels considered varies as the order of the kernel. If the order is chosen to be 2, 16 pixels are considered while if the order is 3, 36 neighbouring pixels are utilized for interpolation. For sufficient image quality, the order is chosen to be 3 in the implementation (Fig. 5).

The Lanczos algorithm gives an output scaled image of the same quality as bicubic but at the cost of higher number of computations as 36 pixels is used for interpolating the value of each output pixel compared to 16 for bicubic. The Lanczos algorithm is comparatively more efficient for multiple scaling operations.

E. Modified Bicubic Image Scaling

According to the bicubic convolution kernel equation, massive calculations of four spots bicubic interpolation algorithm include large cubic and floating point multiplication operation. The bicubic interpolation algorithm discussed above can be slightly modified as shown below (Fig. 6). The co-efficients a0, a1, a2, a3, b0, b1, b2, and b3 are now a function of *dx* and *dy* (the difference between the interpolated co-ordinate and nearest integer source co-ordinate lower than it) instead of the interpolating pixels. This allows us to compute and store the values of the co-efficients for pre-determined values of *dx* and *dy*. This avoids the large number of cubic and floating point operations involved in bicubic interpolation.

$$C[jj] = a0*d0 + a1*d1 + a2*d2 + a3*d3$$

$$Cc = b0*C[0] + b1*C[1] + b2*C[2] + b3*C[3]$$

Where,

$$a0 = -dx/3 + dx*dx/2 - dx*dx*dx/6;$$

$$a1 = 1 - dx/2 - dx*dx/2 + dx*dx*dx/2;$$

$$a2 = dx + dx*dx/2 - dx*dx*dx/2;$$

$$a3 = -dx/6 + dx*dx*dx/6;$$

$$b0 = -dy/3 + dy*dy/2 - dy*dy*dy/6;$$

$$b1 = 1 - dy/2 - dy*dy/2 + dy*dy*dy/2;$$

$$b2 = dy + dy*dy/2 - dy*dy*dy/2;$$

$$b3 = -dy/6 + dy*dy*dy/6;$$

Although the algorithm result has been influenced slightly, computation load is largely simplified. It mainly overcomes the disadvantage that hardware implementation of the algorithm is difficult due to many floating point computations.

The search table modification to the bicubic interpolation algorithm is discussed in [28]. The distance, which the image regards as the unit length l, between two neighbouring pixels in the source image is divided equally into 16 sub-intervals. They are [0,1/16), [1/16,2/16), [2/16,3/16), [3/16,4/16), [4/16,5/16), [5/16,6/16), [6/16,7/16), [7/16,8/16), [8/16,9/16), [9/16,10/16), [10/16,11/16), [11/16,12/16), [12/16,13/16), [13/16,14/16), [14/16,15/16) and [15/16,1). The co-efficients for all 16 values of *dx* and *dy* can be computed beforehand and stored in memory. This reduces the computationally intensive process of determining the co-efficients in real-time. The value of *dx* and *dy* is compared to lie within one of the 16 sub-intervals and the values of coefficients a0, a1, a2, a2, b1, b2, b3, and b4 are chosen accordingly from pre-computed values.

```
float tx = (width_source)/width_dst;
float ty = (height_source)/ height_dst;

for(i=0; i<height_dst; i++)
 for(j=0; j<width_dst; j++)
  {
    x =(int)(tx*j);
    y =(int)(ty*i);

    dx=(float)(tx*j-x);
    dy=(float)(ty*i-y);

    select a0,a1,a2,a3 based on dx and dy
    from pre-computed coefficient  values

    select b0,b1,b2,b3 based on dx and dy
    from pre-computed coefficient  values

    for(jj=0;jj<=3;jj++)
     {
       d0 = P(y-1+jj,x-1);
       d1 = P(y-1+jj,x);
       d2 = P(y-1+jj,x+1);
       d3 = P(y-1+jj,x+2);
       C[jj]= -a0*d0 + a1*d1 + a2*d2 - a3*d3;
     }

   Cc = (-b0*C[0] + b1*C[1] + b2*C[2] - b3*C[3]);
   if(Cc>255) Cc=255;
   if(Cc<0) Cc=0;
   U(i,j) = Cc;
}
```

Figure 6. Modified bicubic image scaling algorithm

IV. Result Analysis

All five algorithms are implemented in C/C++ language. OpenCV libraries are used for image read and write/display only. Dev-C++ (4.9.9.2) software installed on a standard Window XP machine is used for compilation and execution of C/C++ programs. Comparison of computational complexity of all five image scaling algorithms is shown in Table 1. Input test image of size 627x462 and corresponding scaled output images of size 150x150 produced by five implementations are shown in Fig. 7. Fig. 8 shows the input test image of size 609x594 and corresponding scaled output images of size 150x150 produced by five implementations.

Non-adaptive algorithms always face a trade-off between the three image scaling artifacts – edge halo, blurring and aliasing. Nearest neighbour interpolation produces a high aliasing effect resulting in jagged edges. Bilinear interpolation reduces the aliasing effect but causes a moderate blurring of the edges. Bicubic interpolation produces a moderate aliasing, blurring and an edge halo effect. Lanczos interpolation delivers an image quality very similar to that of bicubic. Modified bicubic produces an output of slightly lesser quality as compared to bicubic as the co-efficients are approximated to reduce computational complexity.

Computationally, nearest neighbour interpolation is the least intensive as it considers only one pixel for interpolation. Bilinear uses 4 diagonal pixels for interpolation and therefore, requires more computation than the nearest neighbour method. Bicubic and Lanczos interpolation are computationally very intensive as they require 16 and 36 pixels respectively for interpolating an output pixel value. Lanczos interpolation also utilizes the sine function repeatedly which itself requires a large number of addition and multiplication operations according to Taylor series approximation. Modified bicubic interpolation reduces the number of computations significantly as the co-efficients are computed using search table.

Table 1 : Computational Complexity Comparison

Algorithm	Addition/ Subtractions	Multiplications/Divisions	Sin
Nearest Neighbour	0	2	0
Bilinear	9	12	0
Bicubic	57	69	0
Lanczos	47	118	24
Modified Bicubic	17	24	0

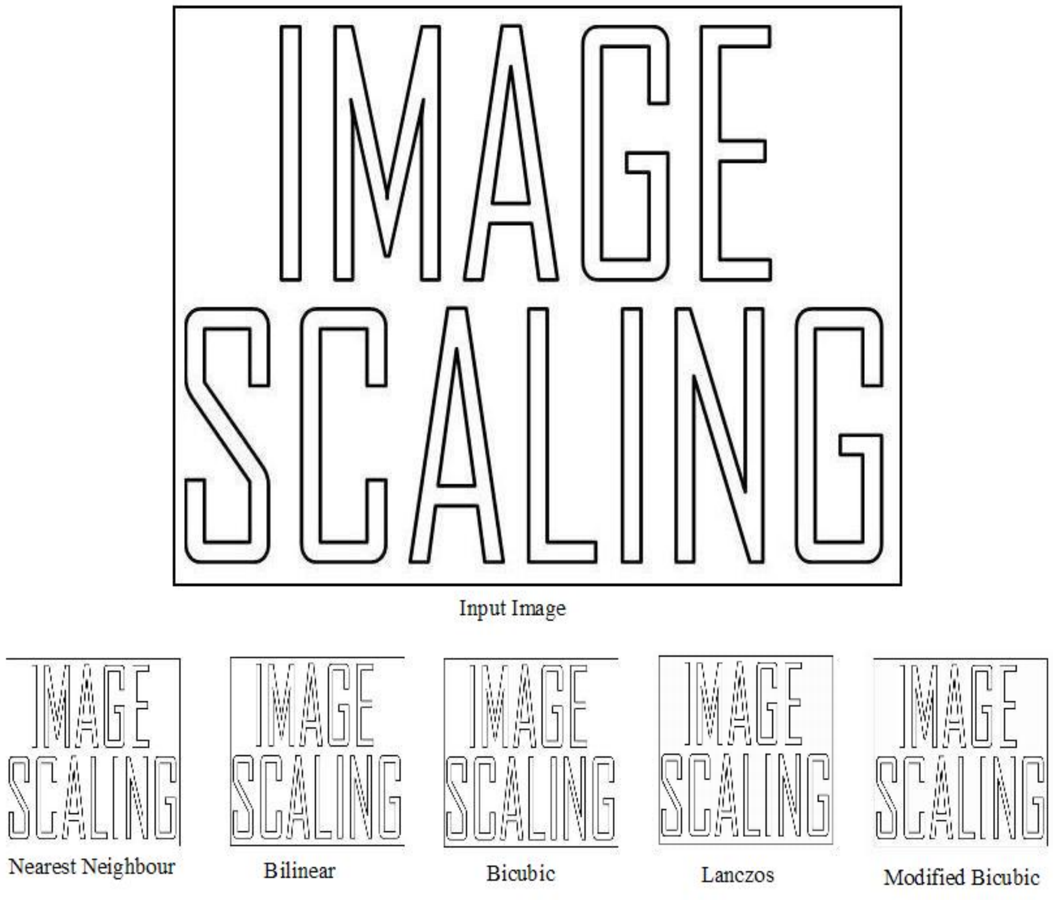

Figure 7. Input test image and corresponding output image for each algorithm

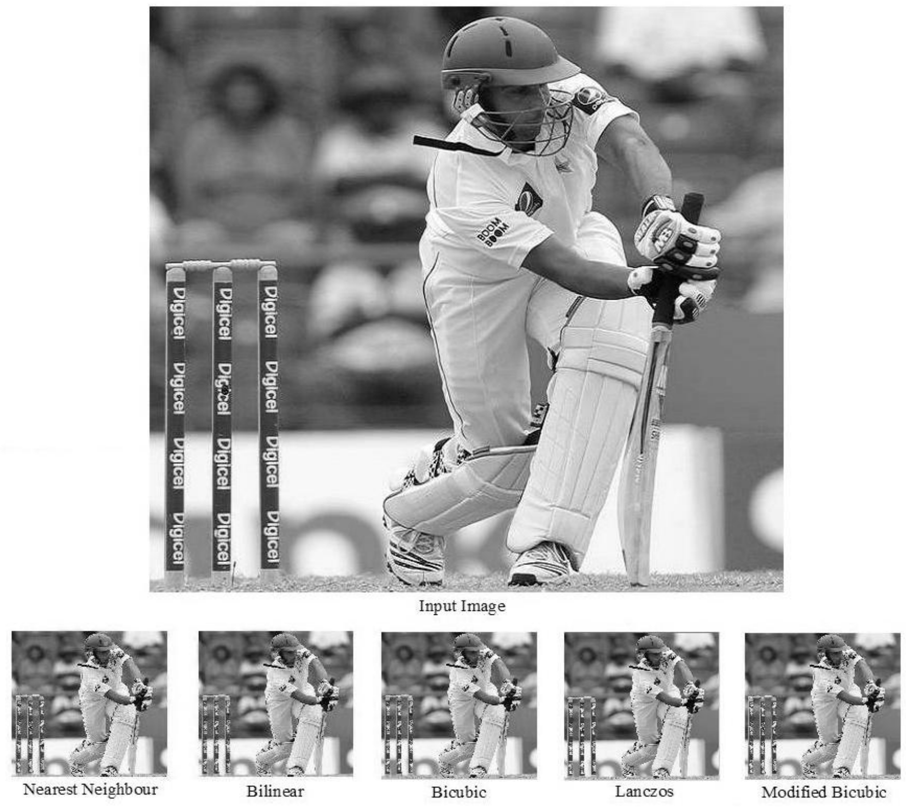

Figure 8. Input test image and corresponding output image for each algorithm

V. Conclusion

Image scaling has become a fundamental processing task and plays an important role in video surveillance applications. Although there are a myriad of interpolation algorithms are available in literature, not all of them are suitable for implementing real-time image scaling. In this paper, we discussed the main non-adaptive image interpolation algorithms suitable for real-time applications. A comparative analysis of five image interpolation algorithms is presented based on the results of their software implementation. Of the five image interpolation algorithms evaluated, the modified bicubic algorithm offers the best trade-off between image quality and computational complexity for real-time image scaling in surveillance applications.

Acknowledgment

The authors express their deep sense of gratitude to Dr. Chandra Shekhar, Director CSIR-CEERI for encouraging research and development activities. Authors would like to thanks Dr. AS Mandal and Raj Singh, Group Leader, IC Design Group, CSIR-CEERI, for their constructive suggestions throughout the year.

References

[1] J. Xiao, X. Zou, Z. Liu, X. Gu, A Novel Adaptive Interpolation Algorithm For Image Resizing, *International Journal of Innovative Computing, Information and Control,* Vol. 3, n. 6(A), pp. 1335-1345, 2007.

[2] Wolberg, G., Massalin, H., *A Fast Algorithm for Digital Image Scaling*, Proceedings of Computer Graphics International (Year of Publication: 1993).

[3] F. Chin, A. Choi, Y. Luo, Optimal Generating Kernels for Image Pyramids by Piecewise Fitting, *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Vol. 14, n. 12, pp. 1190-1198, 1992.

[4] P. Meer, E.S. Baugher, A. Rosenfeld, Frequency Domain Analysis and Synthesis of Image Pyramid Generating Kernels, *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Vol. PAMI-9, n. 4, pp. 512-522, 1987.

[5] P.P.Vaidyanathan, *Multirate Systems and Filter Banks* (Prentice Hall, 1993).

[6] G. Wolberg, *Digital Image Warping* (IEEE Computer Society Press, 1992).

[7] W.H. Press, B.P. Flannery, S.A. Teukolsky, W.T. Vetterling, *Numerical Recipes: The Art of Scientific Computing (FORTRAN Version)* (Cambridge University Press, 1989).

[8] C. Boor, *A Practical Guide to Splines* (Springer-Verlag, 1978).

[9] R.C. Gonzalez, R.E. Woods, *Digital Image Processing* (Prentice Hall, 2007).

[10] K. Turkowski, Filters for common resampling tasks, In A.S. Glassner (Ed.), *Graphic Gems*, 4 (San Diego: Academic Press, 1990, 147-170).

[11] M. Unser, A. Aldroubi, M. Eden, Fast B-Spline Transforms for Continuous Image Representation and Interpolation, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 13, n. 3, pp. 277-285, 1991.

[12] Meijering, E.H.W., Niessen, W.J., Viergever, M.A., *The sinc-approximating kernels of classical polynomial interpolation*, Proceedings of the International Conference on Image Processing (Page: 652-656 Year of Publication: 1999 ISBN: 0-7803-5467-2).

[13] Zhe, W., Jiefu, Z., Mengchu, Z., *A Fast Autoregression Based Image Interpolation Method*, Proceedings of the IEEE International Conference on Networking, Sensing and Control (Page: 1400-1404 Year of Publication: 2008 ISBN: 978-1-4244-1685-1).

[14] Amanatiadis, A., Andreadis, I., Konstatinidis, K., *Fuzzy Area-Based Image Scaling*, Proceedings of the IEEE Instrumentation and Measurement Technology Conference (Page: 1-6 Year of Publication: 2007 ISBN: 1-4244-0588-2).

[15] Mueller, N., Nguyen, T.K., *Image interpolation using classification and stitching*, Proceedings of the IEEE International Conference on Image Processing (Page: 901-904 Year of Publication: 2008 ISBN: 978-1-4244-1765-0).

[16] Morse, B.S., Schwartzwald, D., *Isophote-based interpolation*, Proceedings of the IEEE International Conference on Image Processing (Page: 227-231 Year of Publication: 1998 ISBN: 0-8186-8821-1).

[17] Liang, F., Xie, K., *An Image Interpolation Scheme combined with Artificial Neural Network*, Proceedings of the Third International Conference on Natural Computation (Page: 99-102 Year of Publication: 2007 ISBN: 978-0-7695-2875-5).

[18] S.D. Ruikar, D.D. Doye, Image Denoising using Tri Nonlinear and Nearest Neighbour Interpolation with Wavelet Transform, *International Journal of Information Technology and Computer Science*, Vol.4, n. 9, pp. 36-44, 2012.

[19] D. Su, P. Willis, Image Interpolation by Pixel Level Data-Dependent Triangulation, *Computer Graphics Forum*, Vol.23, n. 2, pp. 189-201, 2004.

[20] Zhang, J., Kuo, C., *Region-adaptive Texture-aware Image Resizing*, Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (Page: 837-840 Year of Publication: 2012 ISBN: 978-1-4673-0045-2).

[21] Jiang, W., Xu, H., Chen, G., Zhao, W., Xu, W., *An Improved Edge-adaptive Image Scaling Algorithm*, Proceedings of the IEEE Eighth International Conference on ASIC (Page: 895-897 Year of Publication: 2009 ISBN: 978-1-4244-3868-6).

[22] Lai, Y., Tzeng, C., Wu, H., *Adaptive Image Scaling Based on Local Edge Directions*, Proceedings of the International Conference on Intelligent and Advanced Systems (Page: 1-4 Year of Publication: 2010 ISBN: 978-1-4244-6623-8).

[23] Jenkins, W., Mather, B., Munson, D., Jr., *Nearest Neighbor And Generalized Inverse Distance Interpolation For Fourier Domain Image Reconstruction*, Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (Page: 1069-1072 Year of Publication: 1985).

[24] W. Ye, A. Entezari, A Geometric Construction of Multivariate Sinc Functions, *IEEE Transactions on Image Processing*, Vol.21, n. 6, pp. 2969-2979, 2012.

[25] Xiao, L., Dong, X., Soong, A.C.K., *Effective Interpolator Design for Pilot Symbol Assisted Modulation Systems*, Proceedings of the IEEE Global Telecommunications Conference (Page: 3671-3675 Year of Publication: 2004 ISBN: 0-7803-8794-5).

[26] Wenbo, T., Jianming, Y., Xiaojin, M., Ji, L., *Power system harmonic detection based on Bartlett–Hann Windowed FFT interpolation*, Proceedings of the Asia-Pacific Power and Energy Engineering Conference (Page: 1-3 Year of Publication: 2012 ISBN: 978-1-4577-0545-8).

[27] Ye, Z., Suri, J., Sun, Y., Janer, R., *Four Image Interpolation Techniques for Ultrasound Breast Phantom Data Acquired Using Fischer's Full Field Digital Mammography and Ultrasound System (FFDMUS): A Comparative Approach*, Proceedings of the IEEE International Conference on Image Processing (Page: 1238-1241 Year of Publication: 2005 ISBN: 0-7803-9134-9).

[28] Zhang, Y., Li, Y., Zhen, J., *The Hardware Realization of Bicubic Interpolation Enlargement Algorithm Based on FPGA*, Proceedings of the Third International Symposium on Information Processing (Page: 277-281 Year of Publication: 2010 ISBN: 978-1-4244-8627-4).

Chetan Suresh is pursuing his undergraduate study in BITS-Pilani, Rajasthan, India. He is a final year student of Electrical and Electronics Engineering. Currently, he is working as a student intern in Cambridge Silicon Radio, Bangalore. His research interests are FPGA Prototyping and Computer Architecture.

Sanjay Singh is working as Scientist in CSIR-Central Electronics Engineering Research Institute, Pilani, Rajasthan, India. He is Member of IEEE - USA, IACSIT - Singapore, and IAENG - Hong Kong. Currently, he is involved in various projects sponsored by Indian Government on Computer Vision and Smart Cameras. His research interests are VLSI architectures for image & video processing algorithms, FPGA Prototyping, and Computer Vision. Prior to joining this research lab, he received his Master in Technology, Master in Electronics, and Bachelor in Science in 2007, 2005, and 2003 respectively from Kurukshetra University, Kurukshetra, Haryana, India. He earned Gold Medal (First Position in University) during his Master in Technology and Master in Electronics. He topped college during his Bachelor in Science. He received more than 20 Merit Certificates and Scholarships during his academic career.

Ravi Saini is working as Scientist in CSIR-Central Electronics Engineering Research Institute, Pilani, Rajasthan, India. He is Member of IEEE – USA. His research interests include VLSI Architectures, ASIC and ASIP Design, HDLs, and FPGA Prototyping. He received his Master in Technology in 2002 from Punjab University, India and Master in Electronics in 2000 from DAVV, Indore, India.

Anil Kumar Saini is working as Scientist in CSIR-Central Electronics Engineering Research Institute, Pilani, Rajasthan, India. He is Member of IEEE – USA and IACSIT – Singapore. His research interests include Analog and Mixed Signal Design, Embedded System Design, and CMOS RF IC design. Prior to joining this research lab, he worked in Cadence, India. He received his Master in Technology and Master in Science in 2003 and 2000 respectively from IIT Roorkee, India.