E-ISSN: 2707-6644
P-ISSN: 2707-6636
www.computersciencejournals.com/ijcpdm
IJCPDM 2021; 2(2): 39-45
Received: 11-05-2021
Accepted: 23-06-2021

PS Parsania
Research Scholar, Sardar Patel University, Vallabh Vidyanagar, Gujarat, India

Dr. PV Virparia
Sardar Patel University, Vallabh Vidyanagar, Gujarat, India

Corresponding Author:
PS Parsania
Research Scholar, Sardar Patel University, Vallabh Vidyanagar, Gujarat, India

Image scaling algorithms using cluster computing

PS Parsania and Dr. PV Virparia

DOI: https://doi.org/10.33545/27076636.2021.v2.i2a.95

Abstract
Images are extensively used in various applications on social media and over the Internet. To view these images on various display devices, image dimensions are changed according to display size. While adjusting an image it goes through the image scaling operation so that image can be properly viewed. To scale an image, one needs to use image scaling algorithm. Various algorithms can be used to determine the new value of the unknown pixel, all these algorithms perform interpolation amongst the nearby pixels in the original image. In this research work, eight image scaling algorithms are used that include Nearest neighbor, Bilinear, Bicubic, Cubic B-Spline, Catmull-Rom, Lanczos order 2, Lanczos order 3, and newly proposed algorithm. The computational time complexity is measured using a single computer and cluster computing for each of these image scaling algorithms and compared with the newly proposed algorithm. The developed algorithm is based on the efficient cubic B-spline function that enhances the image by smooth interpolation at the image edge. Subsequent images generated using various algorithms are then evaluated for their quality using PSNR and UIQI image quality assessment techniques.

Keywords: Image scaling, image interpolation, cluster computing, nearest neighbor, bilinear, bicubic, cubic B-Spline, catmull-rom, lanczos, peak signal to noise ratio (PSNR), Universal image quality index (UIQI)

Introduction
Image scaling is the most common image processing operation in computer graphics. In literature, it is often used with different jargon like image interpolation, image resizing, and image enlarging. Digital images are scaled using various algorithms used by professional software to generate artifacts free images. Most of these software uses well-known image scaling techniques to replicate visually attractive images from the original image. Image scaling is a technique used for resizing an image to a larger or smaller size. This can be achieved by making a new image with high-resolution or low-resolution. Nowadays, we use different display devices like mobile phones, tablets, computer monitors for viewing images from various sources. These images are generally scaled-down (or decreased) to fit on smaller devices or scaled-up (or enlarged) for viewing on a large display size of monitor or television [1]. While scaling up or down an image using image scaling algorithms, it is not possible to discover any more information in the image than already exists, but it recalculates a new pixel value using surrounding pixels. As the image is scaled-up or down, the image quality always reduced that results into many artifacts such as blocky or staircase, ringing, blurring, and edge halo effects [2]. However, there are several such methods of increasing the number of pixels that an image contains, which can be created out of original pixels. These methods are often termed as image interpolation algorithms.
These image interpolation algorithms can be clustered into two different categories. The first one is Non-Adaptive interpolation algorithms and the second is Adaptive interpolation algorithms [3]. In this paper, we used eight different Non-Adaptive interpolation algorithms including one newly proposed algorithm based on the cubic B-spline function. These algorithms generate the target image by calculating the surrounding pixels from the source image with reverse mapping. The logic used to find unknown pixels in a targeted image using these algorithms remains constant, without considering the image properties. These Non-Adaptive algorithms include Nearest-neighbor, Bilinear, Bicubic, Cubic B-spline, Catmull-Rom, Lanczos of order two, Lanczos of order three, and newly proposed image interpolation algorithms named “Optimal Enhanced Cubic B-Spline” (OE-CBS). Depending upon the programming logic used by these algorithms they use 0 to 256 nearby pixels for interpolating a new image [4]. The computational time required to interpolate an image

drastically increase with the increase in the number of pixels considered to scale an image. The image quality generated with more number of surrounding pixels to calculate new pixel results into the better output image. Hence, the number of pixels taken into consideration while scaling the image is directly proportional to the quality of the final output image. As the number of pixels taken into consideration for image interpolation increases, the CPU requires more computational time to generate visually pleasing images [5]. So, the quality of the final image depends upon, how the interpolation algorithm is implemented.

In this paper, we briefly describe various Non-Adaptive image interpolation algorithms and a newly proposed algorithm for Image scaling. These algorithms are implemented on three different types of hardware configuration on a single computer as well as the cluster computing technology with a master and four slave nodes. The images generated using these algorithms are compared for their computational complexity and image quality. Two image quality assessment techniques, the PSNR and UIQI are used to measure image quality.

Image interpolation algorithms

Image interpolation is means by which digital images can be rescaled. Each interpolation technique resulting in a different visual appearance to the output image. However, during the image interpolation, it is better if the image quality is retained and generates the best image without any artifacts during the scaling process [6]. Image interpolation works on rows and columns of the image matrix and tries to achieve the best possible calculation based on the surrounding pixels with new color and intensity in the final image [7]. Figure 1 represents the image scaling for the two-dimensional image.

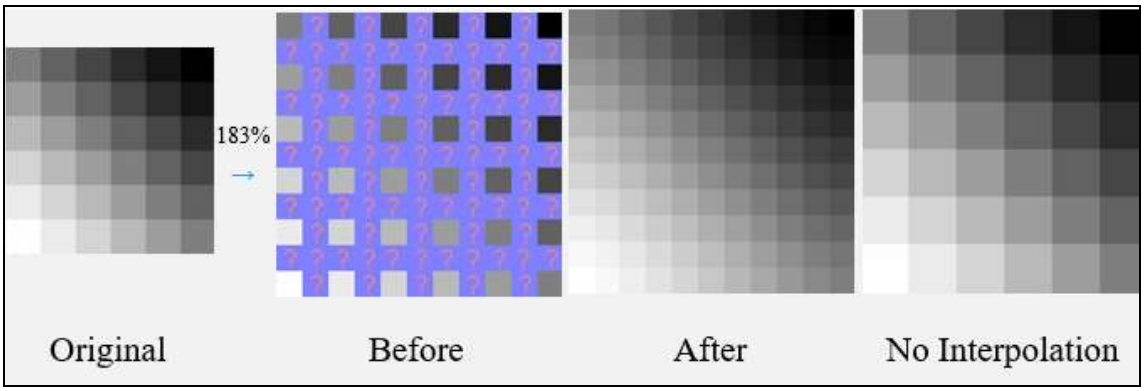

Fig 1: Two dimensional image scaling

Nearest Neighbor interpolation: In this interpolation, it assigns the value of the nearest pixel to the newly calculated pixel in the target image. New pixels is having the same color and intensity as other close-by pixels. The dots of color represented in the original image are replicated to create new pixels in the target image. The computational time required by this method is very less but the resulting image may have artifacts like blocky or jagged edges. This type of interpolation is good for the images with horizontal, vertical, or square patterns otherwise it may not generate eye-pleasing images. This form of interpolation suffers from normally unacceptable effects for both enlarging and reduction of images. The estimated kernel for Nearest Neighbor interpolation is [8]:

$$u(x) = \begin{cases} 0 & |x| > 0.5 \\ 1 & |x| < 0.5 \end{cases}$$

Bilinear interpolation: This interpolation takes the 2-by-2 closest pixels, to take the weighted average of color and intensity to calculate the new pixel value in the target image. If we consider these 4 pixels are positioned at equal distances from each other then the new value for a pixel in the target image is simply the sum of these four pixels divided by four. This method generates a better output image and requires more computation time compared to the nearest neighbor interpolation. The estimated interpolation kernel for bilinear interpolation is [9]:

$$u(x) = \begin{cases} 0 & |x| > 1 \\ 1 - |x| & |x| < 1 \end{cases}$$

Bicubic Interpolation: This interpolation method takes 4-by-4 closest pixels, in a total of 16 pixels to calculate weighted average color and intensity to generate a new pixel in the target image. If the arrangement of these pixels is at various distances from the unknown pixel, closer pixels are given more weighting to calculate a new pixel value. This method generates noticeably sharper images than the Nearest Neighbor and Bilinear methods but requires more computational time. This method is also a standard for image resize in many image editing software. The estimated interpolation kernel for Bicubic interpolation is [10]:

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

Catmull-Rom interpolation: Catmull-Rom interpolation is a family of cubic interpolating splines defined using piecewise polynomial functions that satisfy continuity

between the different points on a line or curve. The mathematically derived formula was initially used for geometric design but has gained its presence in various fields like computer graphics, robotics, and the design of curves and surfaces. This interpolating spline determines the tangent at each point using the previous and next points on the spline. This results into a smooth curve and generates a smoother image particularly at the edges within an image [11].

$$u(x) = 1/2 \begin{cases} 3|x|^3 - 5|x|^2 + 2 & 0<=|x| < 1 \\ -|x|^3 + 5|x|^2 - 8|x| + 4 & 1<= |x|<2 \\ 0 & 2 <=|x| \end{cases}$$

Cubic B-Spline Interpolation*:* This interpolation method also takes 4-by-4 closest pixels to interpolate the new pixel value from the nearest sixteen source pixels with the use of B-spline interpolating functions instead of cubic splines. This method provides quite smooth results at edges compared to the Bicubic method. This interpolation method is having an interesting feature of smoothness that makes it a good option for images with a natural background. The estimated interpolation kernel for Cubic B-Spline interpolation is [12]:

$$u(x) = 1/6 \begin{cases} 3|x|^3 - 6|x|^2 + 4 & 0<=|x| < 1 \\ -|x|^3 + 6|x|^2 - 12|x| + 8 & 1<= |x|<2 \\ 0 & 2 <= |x| \end{cases}$$

Lanczos interpolation

It is derived mathematically for 2-lobed and 3-lobed sinc function and used to prevent aliasing artifacts formation for geometric transformations. This method helps in the smooth interpolation of the pixel value of a digital image between its samples. In this method, each pixel of the given image is translated and scaled to the Lanczos kernel, a form of a windowed low pass sinc filter. The sum of these translated and scaled kernels are then calculated at the given pixel to generate a target image [13]. Lanczos interpolation is the best compromise in terms of reduction of aliasing artifacts for geometric transformations. However, the algorithm requires the most computational time to interpolate an image. The Lanczos interpolation kernel of order n is given by [14]:

$$Ln(x) = \begin{cases} 1 & 0 = |x| \\ \text{sinc}\,(\pi \frac{x}{n})/(\pi \frac{x}{n}) & 0 < |x| < n \\ 0 & n <= |x| \end{cases}$$

OE-CBS interpolation

In order to obtain artifacts-free image quality, we proposed a new algorithm named as "Optimal Enhanced Cubic B-Spline" based on efficient Cubic B-spline function. The goal is to achieve a smooth interpolation curve and to enhance the quality at edges of the image. Our evaluation methodology for the Optimal Enhanced Cubic B-Spline image interpolation algorithm is as follow:

```
SPLINE_INTERPOLATOR (I, x0, y0)

        a← 0
        for i ← 0 to 3 do    \\ iterates over 4 times for rows
                u← | x0| - 1 + i
                b← 0

                for j ← 0 to 3 do    \\ iterates over 4 times for columns
                        v← | y0| - 1 + j
                        b ← b + I (u, v) * W_sp (y0 – v)

                a ←a + b * W_sp (x0 – u)

        return a;
```

Where W_sp is a cubic B-spline piecewise polynomial function as shown below:

$$W\_sp = 1/6 * \begin{cases} (-6a - 9b + 12) * |x|^3 + (6a + 12b - 18) * |x|^2 - 2b + 6 & \text{for } 0<=|x|<1 \\ (-6a - b) * |x|^3 + (30a + 6b) * |x|^2 + (-48a - 12b) * |x| + 24a + 8b & \text{for } 1<=|x|<2 \\ 0 & \text{for } |x|>=2 \end{cases}$$

The constant values a and b in the above equation represent a slope and tangent at a slope of the interpolation function using a = 0.261 and b = 0.379.

Result Analysis

For our research work, we used three different types of hardware configurations to calculate the computational complexity of eight different image scaling algorithms. These algorithms are implemented using ImageJ an image processing software that works on an open-source java programming platform. ImageJ allows creating our own plugin that works on an image or stack of images. These plugins support two different types of interfaces specified by ImageJ. These are the "PlugIn" interface that requires no image to be opened to initialize the user-specific plugin and the "PlugInFilter" interface requires the opened image to be passed as an argument to initialize the user-specific plugin. Our research work uses three images namely Lena, Girl, and Pepper that are downloaded from USC-SIPI Image Database. These images were scaled down to 128 X 128 pixels size and the computational complexity was measured to generate the enlarged images of 256X256, 512X512, 1024X1024, 1280xX1028, 2560X2560, and 6400X6400 pixels using Nearest neighbor, Bilinear, Bicubic, Cubic B-Spline, Catmull-Rom, Lanczos order 2, Lanczos order 3, and newly proposed OE-CBS algorithms.

The computational time complexity of all eight algorithms are measured using Intel® Core i3, Intel® Core i5, and Intel® Core i7, and recorded time in seconds is shown in Table 1, 2, and 3 respectively.

Table 1: Computational time in seconds using Intel® Core i3 CPU @ 2.40 GHz

Algorithms	2X	4X	8X	10X	20X	50X
Nearest Neighbor	0.015	0.028	0.062	0.094	0.312	1.469
Bilinear	0.031	0.047	0.125	0.218	0.797	3.329
Bicubic	0.047	0.172	0.64	0.969	3.922	22.735
Catmull-Rom	0.063	0.218	0.906	1.422	5.516	33.829
Cubic B-Spline	0.068	0.235	0.922	1.438	5.766	34.658
Lanczos 2	0.359	1.813	8.125	13.188	51.112	324.031
Lanczos 3	0.891	4.109	18.048	28.861	112.114	706.799
OE-CBS	0.061	0.214	0.882	1.372	5.464	33.796

Table 2: Computational time in seconds using Intel® Core i5 CPU @ 2.50 GHz

Algorithms	2X	4X	8X	10X	20X	50X
Nearest Neighbor	0.011	0.023	0.045	0.068	0.218	1.282
Bilinear	0.023	0.036	0.094	0.156	0.531	2.814
Bicubic	0.039	0.156	0.442	0.687	2.712	19.023
Catmull-Rom	0.049	0.234	0.536	0.826	3.323	19.612
Cubic B-Spline	0.051	0.256	0.547	0.849	3.368	20.819
Lanczos 2	0.314	1.134	5.853	9.432	42.817	245.275
Lanczos 3	0.608	2.902	12.492	19.926	81.373	515.078
OE-CBS	0.047	0.226	0.519	0.806	3.267	19.536

Table 3: Computational time in seconds using Intel® Core i7 CPU @ 2.70 GHz

Algorithms	2X	4X	8X	10X	20X	50X
Nearest Neighbor	0.008	0.013	0.031	0.046	0.187	0.875
Bilinear	0.015	0.019	0.077	0.093	0.375	2.391
Bicubic	0.028	0.078	0.359	0.562	2.203	13.853
Catmull-Rom	0.033	0.109	0.427	0.667	2.766	17.406
Cubic B-Spline	0.039	0.152	0.462	0.688	2.797	17.456
Lanczos 2	0.203	1.063	4.86	7.886	32.835	203.204
Lanczos 3	0.484	2.375	10.44	16.637	69.514	439.577
OE-CBS	0.031	0.102	0.412	0.646	2.734	17.210

Data reported in Tables 1 to 3 shows that if the output image size increased from 2X to 50X, the corresponding algorithm requires more computational time. The result clearly suggests that if more pixels are to be computed from the input image, the corresponding algorithm requires more computational time. Even the experimental results from the three different hardware configurations clearly show that the computational time decreases with the increase in the processing speed of the CPU and available RAM. The least computational complexity was measured for Nearest Neighbor interpolation and the highest computational complexity was measured for Lanczos order three interpolation algorithm. OE-CBS interpolation algorithm takes 0.061, 0.047, and 0.031second to scale a 128x128 pixel size image to 256x 256 size image on Intel® Core i3, Intel® Core i5, and Intel® Core i7 respectively. Figure 2 shows the input test image (Girl) of size 128X128 pixels and Figure 3 shows the output images of size 256x256 (2X) enlarged using eight different image interpolation algorithms.

Fig 2: Input test image girl (128 X 128) pixels

Fig 3: Output image generated using various image interpolation algorithms

Image quality assessment using PSNR and UIQI algorithms for Girl, Peeper, and Lena images generate the result as shown in Table 4. Experimental result reveals that the PSNR and UIQI values for all three images for Nearest Neighbor interpolation provide the least image quality output while OE-CBS algorithm provides the best image quality output for Pepper and Lena images and second-best for Girl image. Other interpolation algorithms such as Catmull-Rom, Bicubic, Lanczos order 2, and Lanczos order 3 also generate better image quality output.

Table 4: PSNR and UIQI values for girl, peeper, and Lena images

Algorithms	Girl Image		Peeper Image		Lena Image	
	PSNR	UIQI	PSNR	UIQI	PSNR	UIQI
Nearest	27.8073	0.640045	23.8676	0.688558	27.1291	0.716419
Bilinear	31.5117	0.737683	26.0415	0.785411	31.9785	0.818062
Bicubic	31.3207	0.733433	25.7992	0.777105	31.9722	0.817737
Catmull-Rom	31.6477	0.745489	26.0045	0.786335	32.3811	0.834452
Cubic B-Spline	30.3735	0.678875	25.9020	0.770027	30.6273	0.754601
Lanczos 2	31.3658	0.740173	26.3081	0.748318	31.2455	0.765079
Lanczos 3	31.4145	0.733993	25.7603	0.776722	32.3135	0.829707
OE-CBS	31.5664	0.745373	26.3198	0.789282	32.4232	0.837442

The computational complexity of any image scaling algorithm depends upon the programming logic used to implement a specific algorithm. As the implemented algorithm uses different programming logic to scale an image, it takes more time to generate better quality and large size images. Algorithms like Lanczos order two and order three requires much more time to interpolate an image compared to the other algorithms. To make these algorithms work faster we implemented all the eight image interpolation algorithms on the cluster computing technology. For our cluster implementation, we set up a local cluster having one master node and four slave nodes using Windows 10 Pro operating system on each node. The processor and RAM configuration for a master node having Intel® Core i7 CPU @ 3.00 GHz and DDR3 RAM of 32GB

while four slave node computers were configured with Intel® Core i5 CPU @ 3.00 GHz and DDR3 RAM of 8 GB. We used the master slave approach to find the computational time complexity taken by each of these algorithms to scale an image with different sizes. To perform the image scaling using the cluster, the ImageJ macros were implemented on master and slave nodes. The master node performs the slicing of an image and sends each slice to each slave node. Slave nodes scale the image and send the result back to the master node. Once the scaled images from the slave nodes are received, the master node will stitch back these images to form the final image. Table 5 shows the result obtained on cluster computing to enlarge an image with different sizes and Table 6 shows the PSNR and UIQI values for Girl, Peeper, and Lena scaled using cluster computing.

Table 5: Computational time in seconds using cluster computing

Algorithms	2X	4X	8X	10X	20X	50X
Nearest Neighbor	0.991	1.055	1.143	1.269	1.834	5.393
Bilinear	1.004	1.058	1.145	1.326	2.017	5.772
Bicubic	1.020	1.065	1.298	1.440	2.803	10.415
Catmull-Rom	1.027	1.084	1.379	1.512	3.072	12.390
Cubic B-Spline	1.045	1.114	1.386	1.573	3.146	12.405
Lanczos 2	1.082	1.137	2.399	3.114	8.972	50.766
Lanczos 3	1.204	1.737	3.776	4.960	16.590	103.744
OE-CBS	1.051	1.119	1.397	1.590	3.182	13.060

Table 6: PSNR and UIQI using cluster computing

Algorithms	Girl Image		Peeper Image		Lena Image	
	PSNR	UIQI	PSNR	UIQI	PSNR	UIQI
Nearest	27.798	0.6398	23.867	0.6882	27.103	0.7159
Bilinear	31.452	0.7366	26.025	0.7843	31.881	0.8171
Bicubic	31.263	0.7325	25.788	0.7762	31.872	0.8176
Catmull-Rom	31.583	0.7444	25.989	0.7853	32.266	0.8333
Cubic B-Spline	30.342	0.6786	25.890	0.7692	30.579	0.7543
Lanczos 2	31.303	0.7390	26.2910	0.7473	31.153	0.7641
Lanczos 3	31.358	0.7330	25.7464	0.7758	32.202	0.8285
OE-CBS	31.507	0.7414	26.3034	0.7882	32.331	0.8368

The experimental result in Table 5 shows that computational time required to scale an image using cluster computing initially increases compared to the stand-alone computer implementation. This is due to the various additional operation like slicing, uploading, downloading and stitching operation that needs to be performed by cluster. However, the computational time decrease with the increase in output image size. For example, the Lanczos order two and three image interpolation algorithms require 203.204 seconds and 439.577 seconds respectively to interpolate a 50X image on Intel® Core i7 CPU while the same algorithms take 50.766 seconds and 103.744 seconds to scale 50X image using cluster computing.

Table 6 shows Image quality assessment using PSNR and UIQI algorithms for Girl, Peeper, and Lena images. The image quality of scaled images using cluster computing are slightly less compared to the image quality of scaled images using the stand-alone computer as shown in table 4. This is due to the fact that the slices of images are scaled separately on the slave nodes and each algorithm uses different logic to scale the image, so the image border in each sliced image will have a different result when the final image is stitched back on the master node.

Conclusion

The implementation of all the eight algorithms was done with three different hardware configurations and using cluster computing to measure the computational complexity of these algorithms. As the input image size of 128X128 pixels is scaled to 2X, 4X, 8X, 10X, 20X, and 50X sizes, with the increase in the size the more computational time is required to generate the output image.

Estimation of the computational time complexity depends upon several factors ranging from image input size, response time, available memory, clock speed of a computer, and the complexity of each algorithm. Additional operational overheads are required to implement these algorithms on cluster computing. Computationally efficient algorithms such as Nearest Neighbor and Bilinear provide poor image quality. The combination of computational complexity and image quality makes the interpolation algorithms like Bicubic, Catmull-Rom, cubic B-Spline, and OE-CBS suitable for image scaling operation that takes slightly more computational time but generates visually better quality images than Nearest Neighbor and Bilinear algorithms. Lanczos order two and three image interpolation algorithms generate visually good images but the computational time is very high to interpolate the image and the implementation is not suitable for real-world applications.

References

1. Deshpande A, Patavardhan PP. Survey of super resolution techniques. ICTACT J Image Video Process. 2019;9(1):1784-1792.
2. Chen G, Zhao H, Pang CK, Li T, Pang C. Image scaling: How hard can it be? IEEE Access. 2019;7:129452-129465.
3. Htwe AN, Nge MM. Image resampling using non-adaptive interpolation. In: Proceedings of the Fourth Local Conference on Parallel and Soft Computing; Yangon, Myanmar; c2009 Nov.

4. Parsania PS, Virparia PV. Computational time complexity of image interpolation algorithms. Int J Comput Sci Eng. 2018;6(4):491-496.
5. Khan S, Lee DH, Khan MA, Gilal AR, Mujtaba G. Efficient edge-based image interpolation method using neighboring slope information. IEEE Access. 2019;7:133539-1335348.
6. Kherd A, Saaban A, Eskandar I. Image enlargement using biharmonic DP-ball surface. In: Proceedings of the 2019 First International Conference of Intelligent Computing and Engineering (ICOICE). Kuala Lumpur, Malaysia; c2019 Dec. p. 1-4.
7. Moses CJ, Selvathi D, Sophia VM. VLSI architectures for image interpolation: a survey. VLSI Design; c2014. p. 11-17.
8. Shah K, Pandya J, Vahora S. A survey on super resolution image reconstruction techniques. Int J Eng Res Technol. 2013;2(4):1897-1901.
9. Kaur G, Kaur J. A comparative study of image demosaicing. Int J Comput. Sci. Eng. 2015;3(7):98-102.
10. Sinha A, Kumar M, Jaiswal AK, Saxena R. Performance analysis of high resolution images using interpolation techniques in multimedia communication system. Signal Image Process. 2014;5(2):39-49.
11. Wu T, Bai B, Wang P. Parallel Catmull-Rom spline interpolation algorithm for image zooming based on CUDA. Int J Appl Math Inform Sci. 2013;7(2):533-537.
12. Han D. Comparison of commonly used image interpolation methods. In: Proceedings of the 2nd International Conference on Computer Science and Electronics Engineering Hangzhou, China. p. [pages missing]. Atlantis Press; c2013 Mar.
13. Sinha A, Kumar M, Jaiswal AK, Saxena R. Performance analysis of high resolution images using interpolation techniques in multimedia communication system. Signal Image Process. 2014;5(2):39-49.
14. Tai SC, Kuo TM, Li KH. An efficient super resolution algorithm using simple linear regression. In: Proceedings of the 2013 Second International Conference on Robot, Vision and Signal Processing Hong Kong; c2013 Dec. p. 287-290.