ISSN 0002-306X. Proc. of the RA NAS and NPUA Ser. of tech. sc. 2020. V. LXXIII, N2

UDC 004.932.4

COMPUTER SCIENCE
AND INFORMATICS

E.H. DANOYAN, H.Y. AYUNTS

IMAGE INTERPOLATION USING MIXED SPLINES

In our digital era image interpolation and resampling have many usages in various fields (computer graphics, medicine, geography, astronomy). This paper presents an overview of some interpolation techniques (nearest neighbor, bilinear, bicubic) and proposes a new interpolation type where the results are good and the execution speed is fast.

Keywords: Image Interpolation, Image Resampling, Bicubic, Bilinear, Gradient, Sobel.

Introduction. Image resampling is a fundamental and widely used image operation, with applications in image display, compression and progressive transmission. Upsampling is the increasing of the spatial resolution while keeping the 2D representation of an image, that is improving the quality of an image by increasing the number of pixels. Downsampling is the reduction of the resolution. It is used to reduce the storage and for image transmission. The standard methods for resampling are the nearest neighbor, bilinear interpolation, bicubic interpolation. These methods have been thoroughly studied and compared in the past [1]. Bicubic gives better results compared with the other two, but it also has the longest execution time [2]. Therefore, it is one of the most widely used interpolation techniques. In this paper, we examine these methods and propose a new interpolation type (mixed interpolation) which gives good results and faster execution time.

1. Image Interpolation Algorithms. The main method for image resampling is image interpolation. There are three main interpolation techniques:

- Nearest neighbor;
- Bilinear interpolation;
- Bicubic interpolation.

1.1. Nearest Neighbor. The nearest neighbor (Fig. 1) is the simplest interpolation type. The color of a pixel in the result image is the color of the nearest pixel of the original image. From [3], the interpolation kernel is:

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

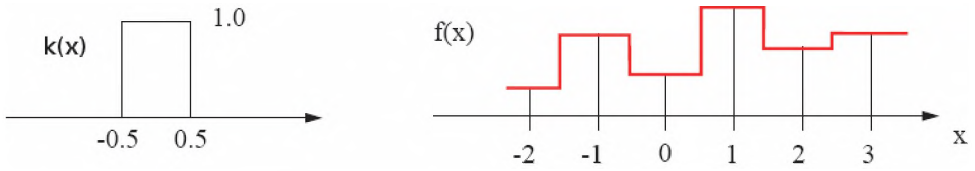

Fig. 1. Nearest neighbor interpolation

This method produces blocky effects on the result in case of upsampling and removes thin edges in case of downsampling. However it is the fastest algorithm.

1.2. Bilinear Interpolation. Bilinear interpolation (Fig. 2) considers the closest $2x2$ neighborhood of the known pixel values surrounding the unknown pixel. From [3], the interpolation kernel is:

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

This method gives better results than the last one.

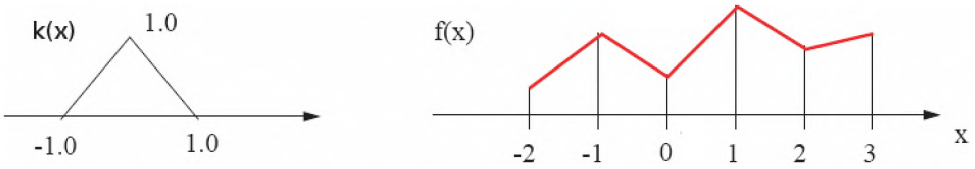

Fig. 2. Bilinear interpolation

1.3. Bicubic Interpolation. Bicubic interpolation (Fig. 3) considers the closest $4x4$ neighborhood of the known pixels. The convolution kernel is composed of cubic splines. The kernel is given with this formula from [4]:

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

where $a \in [-0.75; -0.5]$.

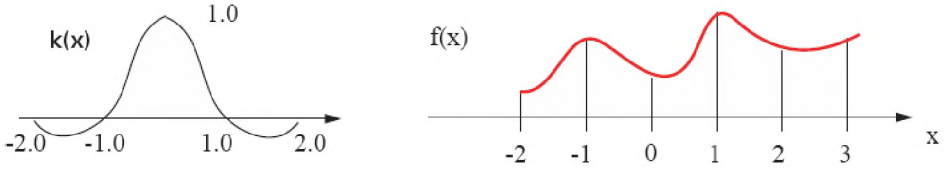

Fig. 3. Bicubic interpolation

In all these methods, the result value of the unknown pixel is calculated with this formula from [5]:

$$g(x,y) = \sum_{x_0=[x]-[\frac{m}{2}]}^{[x]+[\frac{m}{2}]+1} \sum_{y_0=[y]-[\frac{m}{2}]}^{[y]+[\frac{m}{2}]+1} k(x-x_0)k(y-y_0)f(x_0,y_0),$$

where $m = 0$ for the nearest neighbor method, $m = 1$ for bilinear and $m = 3$ for bicubic interpolation.

The results show that bicubic is one of the best interpolation techniques but has the longest execution time [2], because for every unknown pixel, the value of 16 other pixels need to be considered. On the other hand, it gives smoother results in case of upsampling and less loss of information in case of downsampling.

2. The proposed method: Mixed Interpolation. The higher the interpolation order gets, the better are getting the results, though the execution time is getting slower and slower. Image resizing is used in video processing where every millisecond of working time is important. So we propose the adaptive mixed interpolation method from these three main methods. We will use the image gradient definition from [5].

The first derivatives in image processing are implemented, using the magnitude of the gradient. For the function $f(x,y)$, the gradient of f at coordinates (x,y) is defined as a two-dimensional column vector:

$$\nabla f \equiv grad(f) \equiv \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}.$$

The magnitude (length) of vector ∇f is denoted as $M(x,y)$, where

$$M(x,y) = mag(\nabla f) = \sqrt{{g_x}^2 + {g_y}^2}$$

is the value at (x,y) of the rate of change in the direction of the gradient vector. $M(x,y)$ is an image of the same size as the original. Usually, we refer to it as the gradient image.

We now define discrete approximations to the preceding equations and hence formulate the appropriate filter masks [5]. In order to simplify the discussion that follows, we will use the notation in Fig. 4(1) to denote the intensities of image points in a 3x3 region. For example, the center point (z_5), denotes $f(x,y)$ at an arbitrary location, (x,y), z_1 denotes $f(x-1,y-1)$ and so on. Approximations to g_x and g_y are as follows:

$$g_x = \frac{\partial f}{\partial x} = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3),$$

$$g_y = \frac{\partial f}{\partial y} = (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7).$$

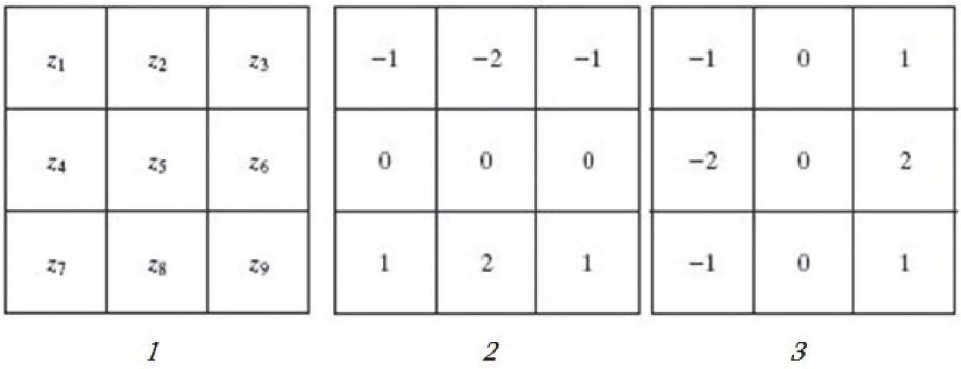

Fig. 4. A 3x3 region of an image (1), Sobel operators (2), (3)

These equations can be implemented using Sobel operators from Fig. 4 (2), (3). The difference between the third and first rows of the 3x3 image region implemented by the mask in Fig. 4(2) approximates the partial derivative in the x-direction, and the difference between the third and first columns in the other mask approximates the derivative in the y-direction.

The idea behind using a weight value of 2 in the center coefficient is to achieve some smoothing by giving more importance to the center point. After computing the partial derivatives with these masks, we can approximate the magnitude of the gradient.

$$M(x,y) \approx |(z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)| + |(z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)|.$$

After getting the gradient image we divide the intensity range into three parts:

$$[\mathrm{m}, \mathrm{M}] \rightarrow \begin{cases} [0, \mathrm{a}] \rightarrow \text{nearest neighbor}, \\ (\mathrm{a}, \mathrm{b}] \rightarrow \text{bilinear}, \\ (\mathrm{b}, \mathrm{M}] \rightarrow \text{bicubic}, \end{cases}$$

where $m = \min_{(x,y)} M(x,y)$, $M \equiv \max_{(x,y)} M(x,y)$ and $\mathrm{m} < a < b < M$ (for example: $\mathrm{a} = \left[\mathrm{m} + \frac{(\mathrm{M}-\mathrm{m})}{3}\right]$, $\mathrm{b} = \left[\mathrm{m} + \frac{2(\mathrm{M}-\mathrm{m})}{3}\right]$).

If the $2x2$ neighborhood of the known pixels surrounding the unknown pixel has a small intensity of the gradient, we will use the nearest neighbor method on that pixel. Likewise we will use bilinear interpolation for medium values. Otherwise, we will use bicubic interpolation. So we can decrease the execution time by using the fastest interpolation for the areas that don't have to be interpolated very precisely.

3. Results and Discussion. In order to test the performance of the algorithms, we have developed a C++ implementation on Intel(R) Core(TM) i3-5005U CPU 2.00GHz, 4.0GB RAM computer [6].

Here are some upsampling results of different-sized images ($50x50$, $100x56$, $100x45$), where the scale factor is 4 (Fig. 5, Fig. 6 and Fig. 7).

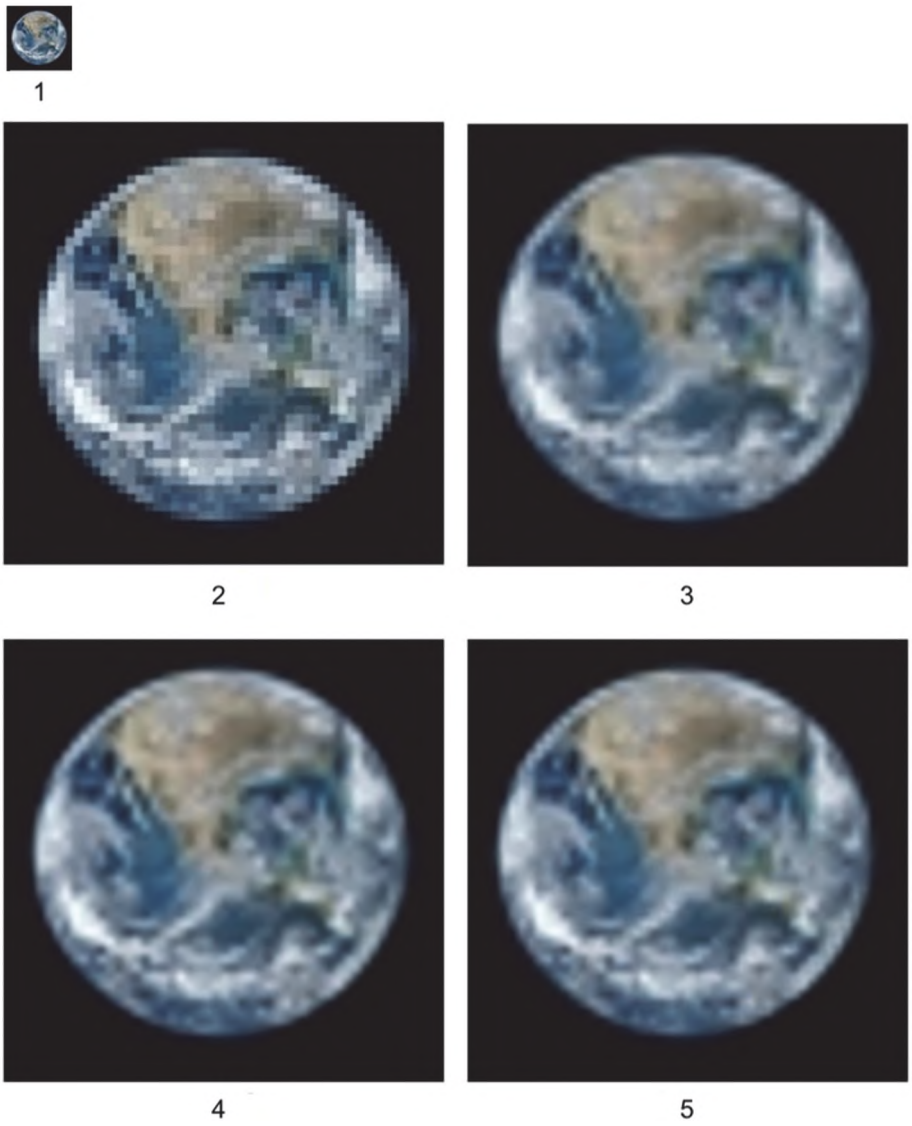

Fig. 5. Source image (1), nearest neighbor interpolation (2), bilinear interpolation (3), bicubic interpolation (4), mixed interpolation (5)

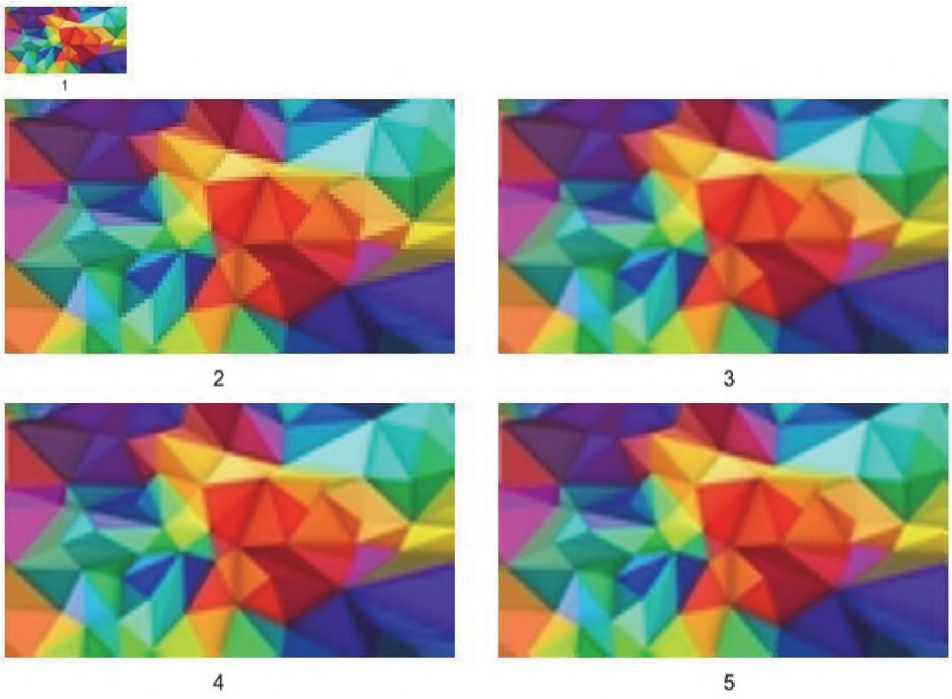

Fig. 6. Source image (1), nearest neighbor interpolation (2), bilinear interpolation (3), bicubic interpolation (4), mixed interpolation (5)

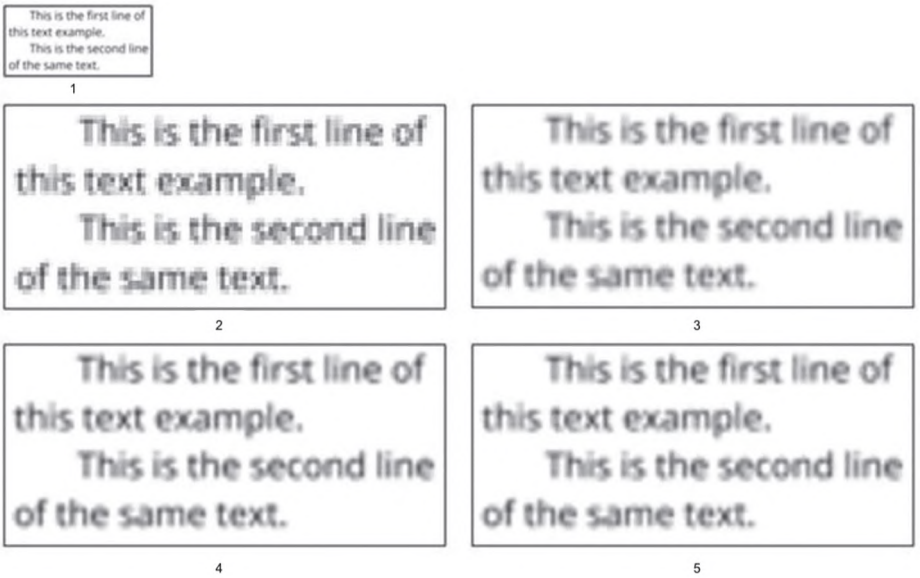

Fig. 7. Source image (1), nearest neighbor interpolation (2), bilinear interpolation (3), bicubic interpolation (4), mixed interpolation (5)

Table

Timing Results (milliseconds)

Method / Scale	Nearest neighbor		Bilinear		Bicubic		Mixed	
2	40	12	300	82	1068	270	483	168
4	161	34	1676	303	4695	1185	2494	663
8	605	180	6183	1409	18864	4196	9301	2519

In the Table above are shown the timing results of $200x100, 100x50$-sized image upsampling. The results show that the mixed interpolation can give as good results as the bicubic, working 1.5-2 times faster. So, for example, in the image of the earth (Fig. 5), there is no need to calculate the pixel values from the 16 neighbor pixels in the black areas. The color of the pixel will be black in all methods. That is why, we use faster methods in these areas, decreasing the execution time.

Image resizing has many applications in various fields, such as computer graphics, medicine, geography, astronomy. In all these cases, good results and fast time is very important especially when the program is working on a large number of images. The best example is the usage in video processing where we can consider video as a large number of frames.

Conclusion. Mixed image interpolation is the overlay of the three other methods for getting good results and faster execution time. This method can be used in various fields where the working time of the algorithm is very important, and the results will have to be as good as possible for the corresponding time.

REFERENCES

1. **Parker J.A., Kenyon R.V. and Troxel D.E.** Comparison of Interpolating Methods for Image Resampling // IEEE Transactions on Medical Imaging. - 1983. - Vol. MI-2. - P. 31-39.
2. **Sane P.S., Gavade A.B.** Super Resolution Image Reconstruction by using Bicubic Interpolation // Advanced Technologies in Electrical and Electronic Systems. – 2013. - P. 204-209.
3. **Roy R., Pal M. and Gulati T.** Zooming Digital Images using Interpolation Techniques // International Journal of Application or Innovation in Engineering & Management. – 2013. - Vol. 2, issue 4. - P. 34-45.

4. **Keys R.G.** Cubic Convolution Interpolation for Digital Image Processing // IEEE Transactions on Acoustics, Speech, and Signal Processing. – 1981.- Vol. ASSP-29. - P. 1153-1160.
5. **Gonzalez R.C., Woods R.E.** Digital Image Processing. - 3rd edition. – 2008. - 976 p.
6. https://github.com/HrachA/ImageInterpolation

Yerenav State University. The material is received on 03.12.2019.

Է.Հ. ԴԱՆՈՅԱՆ, Հ.Ց. ԱՅՈՒՆՑ

ՊԱՏԿԵՐՆԵՐԻ ԻՆՏԵՐՊՈԼՅԱՑԻԱՆ ԽԱՌԸ ՍՊԼԱՅՆՆԵՐԻ ՄԻՋՈՑՈՎ

Ներկա տեխնոլոգիական դարաշրջանում պատկերների ինտերպոլյացիան ունի շատ կիրառություններ տարբեր բնագավառներում (համակարգչային գրաֆիկա, բժշկություն, աշխարհագրություն, աստղագիտություն): Ներկայացվում են արդեն հայտնի ինտերպոլյացիայի եղանակները (ամենամոտ հարևան, երկխորանարդային, երկգծային), և առաջարկվում է ինտերպոլյացիայի նոր եղանակ, որը տալիս է լավ արդյունքներ և ծրագրի արագ աշխատանք:

Առանցքային բառեր. պատկերների ինտերպոլյացիա, պատկերների մասշտաբավորում, երկխորանարդային, երկգծային, գրադիենտ, Սոբել:

Э.А. ДАНОЯН, Г.Ю. АЮНЦ

ИНТЕРПОЛЯЦИЯ ИЗОБРАЖЕНИЙ СМЕШАННЫМИ СПЛАЙНАМИ

В настоящее время наряду с технологическим прогрессом интерполяция изображений получила широкое применение в различных областях (компьютерная графика, медицина, география, астрономия). Представлен обзор некоторых методов интерполяции (ближайший сосед, билинейный, бикубический). Предлагается новый метод интерполяции, дающий удовлетворительные результаты и высокую скорость выполнения программы.

Ключевые слова: интерполяция изображений, масштабирование изображений, бикубический, билинейный, градиент, Собель.