Research of the effectiveness of parallel multithreaded realizations of interpolation methods for scaling raster images

A A Vnukov¹, M B Shershnev¹

¹National research University Higher school of Economics, 20 Myasnitskaya ulitsa, Moscow 101000 Russia

Abstract. The aim of this work is the software implementation of three image scaling algorithms using parallel computations, as well as the development of an application with a graphical user interface for the Windows operating system to demonstrate the operation of algorithms and to study the relationship between system performance, algorithm execution time and the degree of parallelization of computations. Three methods of interpolation were studied, formalized and adapted to scale images. The result of the work is a program for scaling images by different methods. Comparison of the quality of scaling by different methods is given.

1. Introduction

These Scaling of images is used in a wide range of applied problems. This is one of the most demanded operations in computer graphics. [1]

Interpolation methods are widely used to solve various mathematical problems. There are many methods of interpolation. The most popular are the following image interpolation methods: bicubic interpolation, bilinear interpolation, stepwise interpolation (nearest-neighbor interpolation). These methods are listed in order of decreasing computational complexity and, accordingly, in order of decreasing the quality of the resulting image. [2]

The speed of execution of interpolation algorithms directly depends on the possibilities of their practical application. Often you need to scale images in real time, for example, when playing video. To effectively use the computing resources of modern multiprocessor computers, you need to use multi-threaded computing. [3]

2. Overview of existing methods of interpolation of images

When interpolating images, the image itself plays the role of an interpolated function, the value of which is the color of the pixel. Image pixels are points where the value of the function (pixel color) is known. Interpolation contains intermediate values of the function (colors of new pixels), the image does not necessarily increase, the interpolation methods allow to arbitrarily change the image size and aspect ratio.

2.1 Nearest neighbor method

Stepwise interpolation (also known as the nearest neighbor method) is the simplest method of multidimensional interpolation. The nearest neighbor algorithm chooses the nearest point value and does not take into account the values of neighboring points at all. The algorithm is very simple to implement and widely used.
In Figure 1, the red dots represent the input data, the points at which the value of the interpolated function is known. The lines of blue denote the interpolation function. The stepwise interpolation algorithm is the simplest, both in terms of implementation and in terms of computational complexity. However, when the image is enlarged by this method, the so-called staircase effect is observed.

![Figure 1. Interpolant function for nearest neighbour method.](image1)

2.2 **Bilinear interpolation**

In the case of linear interpolation, in order to find intermediate values, a piecewise linear function is constructed from known values. Bilinear interpolation is used for functions of two variables: first, linear interpolation is performed for each of the measurements, thus two auxiliary points are calculated, then linear interpolation between these auxiliary points is performed.

The formulas for linear and bilinear interpolation are:

\[ f_{\text{linear}}(p_0, p_1, x) = x(p_1 - p_0) + p_0 \]

\[ f_{\text{bilinear}}(p_{00}, p_{01}, p_{10}, p_{11}, x, y) = f_{\text{linear}}(f_{\text{linear}}(p_{00}, p_{01}, x), f_{\text{linear}}(p_{10}, p_{11}, x), y) \]

\[ p_{00}, \ldots, p_{11} \] – the colors of the four nearest pixels.

\[ x, y \] – the relative coordinates of the point at which the color is calculated.

The bilinear interpolation algorithm is computationally more complex than the stepwise interpolation algorithm, but the resulting image looks much smoother than the result of using the stepwise interpolation method. This algorithm is often used in graphic editors, as it allows to obtain acceptable in most cases quality, without requiring a large computing power. However, if the original image is enlarged many times with this algorithm, unnatural sharp color transitions can occur in some places of the resulting image.

![Figure 2. Interpolant function for bilinear interpolation.](image2)
2.3 **Bicubic interpolation**

Bicubic interpolation is the extension of one-dimensional interpolation by polynomials of the third degree to the case of a function of two variables. The surface obtained as a result of bicubic interpolation is a smooth function, unlike surfaces obtained as a result of bilinear interpolation or interpolation by the nearest neighbor method.

Bicubic interpolation assumes that the interpolated function is given not only by its values at certain points, but also by the values of the derivative at these points.

In the case of image interpolation, it plays the role of an interpolated function, whose known values are the colors of the pixels of the original image, the values of the derivative are not known. You can assume the derivative value to be zero, but you can use the slope of the line between the previous and next point as a derivative to get a smoother look.

The formulas for cubic and bicubic interpolation are:

\[
f_{\text{cubic}}(p_0, \ldots, p_3, x) = x^3 \left( -\frac{1}{2} p_0 + \frac{3}{2} p_1 - \frac{3}{2} p_2 + \frac{1}{2} p_3 \right) + x^2 \left( p_0 - \frac{5}{2} p_1 + 2 p_2 - \frac{1}{2} p_3 \right) + x \left( -\frac{1}{2} p_0 + \frac{1}{2} p_2 \right) + p_1
\]

\[
f_{\text{bicubic}}(p_{00}, \ldots, p_{33}, x, y) = f(f_{\text{cubic}}(p_{00}, \ldots, p_{03}, x), f_{\text{cubic}}(p_{10}, \ldots, p_{13}, x), f_{\text{cubic}}(p_{20}, \ldots, p_{23}, x), f_{\text{cubic}}(p_{30}, \ldots, p_{33}, x), y)
\]

Where \( p_{00}, \ldots, p_{33} \) – the color values of the 16 nearest pixels surrounding the calculated value.

\( x, y \) – the relative coordinates of the point at which the color is calculated.

The bicubic interpolation algorithm is computationally more complex than the bilinear interpolation algorithm, however the resulting image looks smoother.

![Figure 3. Interpolant function for bicubic interpolation.](image)

3. **Parallelizing an algorithm**

Raster images often consist of a large number of pixels of the order of several million. Interpolation of such images is associated with a large number of calculations. At the same time, new pixels can be calculated independently of each other, so that the efficiency of the algorithm can be significantly increased by parallelizing the computations.

Algorithms of interpolation considered in this paper can be parallelized according to the following scheme: the image is divided into segments by rows. Each thread processes its part of the image, then the parts are combined to form the resulting image. Segmentation by line is important, since in the
computer's memory the image is stored as a string sequence, and sequential memory access operations are faster than random access operations.

4. Create an application
The Main features:
- Open and save files with images in BMP format
- Display an editable image in the application window
- Scale images using the user-specified parameters: interpolation type, degree of scale change for each direction, number of streams used.

The main task of the application is to demonstrate the work of realized image interpolation algorithms. The application should simplify the process of testing algorithms and make it more visible. The interface is represented by the following elements: the main window in which the image is displayed, the main menu for accessing functions, the dialogs for opening and saving a file, and the scaling dialog in which the user can specify the scaling parameters. Also, the program displays the amount of time spent processing the image.

5. Performance Comparison and Scalability Research
Graph 1 shows the results of measurements of the image processing time when the algorithm is parallelized to a different number of streams for each of the interpolation methods under consideration. On the vertical axis there is time, on the horizontal axis - the number of streams. The graph shows that the bilinear interpolation algorithm works on average 2.4 times longer than the stepwise interpolation algorithm, and the bicubic interpolation algorithm is 4.5 times longer than the bilinear interpolation algorithm. Parallelization has resulted in a 300% - 400% increase in productivity.

6. Examples of image scaling

6.1 Example 1
The original image has a size of 2x2 pixels. It was increased 100 times on both sides. The result of the three methods is shown in Figure 4.
Stepwise interpolation does not change the structure of the image. Four areas are clearly visible, each of which corresponds to one of the four pixels of the original image. Bilinear interpolation makes the color transitions smooth, but in some places unnatural vertical and horizontal bands are noticeable. Bicubic interpolation gives an absolutely smooth image with smooth color transitions.

6.2 Example 2
The original image was enlarged 100 times on both sides. The difference in interpolation quality is clearly visible if you look at the yellow dot in the upper left corner. In the case of bilinear interpolation, the vertical and horizontal lines in the center are expressed inside the point. With the increase using bicubic interpolation, one can see an almost ideal circle with smooth color transitions at the boundary.

7. Conclusion
In the course of this work, three parallel algorithms for scaling raster images by different interpolation methods were implemented. As a result of the work, an application with a graphical user interface was created, which allows to scale images by different interpolation methods and to save the result. The application allows you to compare different interpolation methods for the quality of image processing.

In the course of testing, the assumption of the feasibility of parallelizing the bicubic interpolation algorithm was confirmed. The use of several calculation flows provides a significant acceleration of the algorithm.

8. References
[1] G. Keys Robert, "Cubic Convolution Interpolation for Digital Image Processing", IEEE Transactions on Acoustics, Speech and Signal Processing, vol. 29, no. 6, 1981, pp. 1153-1160.
[2] Meijering Erik, Unser Michael, "A Note on Cubic Convolution Interpolation", IEEE Transactions on Image Processing, vol. 12, no. 4, April 2003, pp. 477-479.
[3] Vnukov A. A., Egorov I. V. "Investigation of the effectiveness of software implementation of the multithreaded scaling algorithm by the method of bilinear interpolation", Mining Information and Analytical Bulletin (scientific and technical journal). 2014. № 9. C. 264-271.