Adaptive Interpolation Algorithm for Real-time Image Resizing

Jianping Xiao*, Xuecheng Zou, Zhenglin Liu, and Xu Guo
Department of Electronic Science and Technology ,
Huazhong University of Science and Technology
430074,Wuhan, China
** xjp_csut@21cn.com*

Abstract

In this paper, an adaptive interpolation algorithm is presented based on the Newton Polynomial to improve the limitation of the traditional algorithm for image resizing. The second-order difference of adjacent pixels gray values shows the relativity among the pixels. Accordingly, the adaptive function for image interpolation is deduced according to both this relativity and the classical Newton polynomial. Then the efficiency of our method is compared with that of the traditional algorithm for image resizing in Matlab. Furthermore, the implementation circuit architecture is devised by three stage paralleling pipelines for the adaptive image resizing algorithm and is verified in FPGA (field programmable gate array). The experimental results show that our proposed algorithm excels the bicubic interpolation in visual effect, and has a lower complexity. Therefore, the algorithm adapts to real-time image resizing.

1. Introduction

Image resizing widely applied in numerous fields such as medical image processing, military applications and consumer electronics [1]. For instance, we have to enlarge images in HDTV or medical image display, or a scale-down image will fit the mini-size LCD panel in portable instruments. The efficiency of any image resizing algorithm is determined by two main factors: the quality of the obtained image and the computational complexity.

The common approaches for image enlargement are the nearest neighbor, the bilinear and the bicubic interpolations. However, the magnified images processed by these techniques often appear "blocky" or blurring. To avoid the shortcomings of these methods, researchers have proposed various algorithms for image resizing. The visual effect of the target image is improved to various extents. For instance, Huang and Chen [2] proposed a hybrid interpolation filtering method, Unser et al. [3] developed more accurate spline interpolation algorithms, and Jensen and Anastassiou [4] used an edge fitting model to enlarge images, respectively. However, these methods are not fit for real-time image resizing because of their complexity.

In this paper, an adaptive interpolation algorithm is proposed based on Newton polynomial for real-time image resizing. According to the relative positions of the target and source pixels, two groups of pixels in the source image correspond to every target pixel. And one of the two groups is automatically selected by comparing their second-order difference values. Then the gray value of the corresponding target pixel is computed with second-order Newton interpolation function. As a result, The PSNR (peak signal noise ratio) of our proposed method is higher than that of the bicubic interpolation and achieves pleasant visual appearance. Furthermore, it adapts to implement by hardware for its low complexity. This paper is organized as follows: In section 2, we explain the mathematical principle on adaptive image resizing based on the Newton polynomial. The implementation architecture for our proposed algorithm is devised in section 3. In section 4, the simulation and experimental results of various interpolation algorithms are given. The conclusion is summarized in section 5.

2. Theory of the adaptive Newton interpolation

2.1. Newton interpolation function

Given that the value of function $f(x)$ on the equally spaced nod x_i is shown by the equation $f(x_i) = f_i$ $(i=0,1,\ldots,n)$, we define that

$$\Delta f_i = f_{i+1} - f_i \qquad (1)$$

is the first-order difference, and that k-th order differrence is

$$\Delta^k f_i = \Delta^{k-1} f_{i+1} - \Delta^{k-1} f_i \qquad (2).$$

The n-th order Newton interpolation function shows as follows:

$$N_n(x_0 + th) = \sum_{k=0}^{n} \frac{\Delta^k f_0}{k!} \prod_{j=0}^{k-1} (t - j) \qquad (3).$$

where, $t = \frac{x - x_0}{k}$, *and* h *is step.*

Two-dimension digital image interpolation can be decomposed into two one-dimension interpolations. First, the discrete image signals are interpolated in horizontal direction. Second, the horizontal interpolation results are interpolated in vertical direction, where the step h is 1.

When $n=1$, equation (3) is the bilinear interpolation function. And when $n=2$, equation (3) becomes the second-order Newton interpolation:

$$N_2(x_0 + t) = f_0 + \Delta f_0 t + \frac{\Delta^2 f_0}{2!} t(t-1) \qquad (4).$$

When $n=3$, the third-order function is:

$$N_3(x_0 + t) = f_0 + \Delta f_0 t + \frac{\Delta^2 f_0}{2!} t(t-1) + \frac{\Delta^3 f_0}{3!} t(t-1)(t-2) \qquad (5).$$

To process the image resizing, the pixel sequence in a certain direction (such as horizontal direction) of the source image is regarded as a group of equidistance nods, thus any target pixel's gray value in the same direction can be computed by some source pixels by interpolation function. If the third-order Newton interpolation function is adopted (see figure 1), except for border pixels in the source image, any target pixel N is corresponding to a source pixel group including f_0, f_1, f_2, f_3. Then the gray value of target pixel N can be computed by equation (5), where t is the distance between N and f_0, $(1 \le t < 2)$.

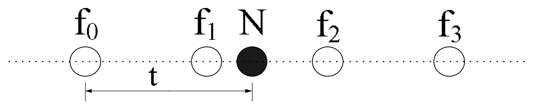

Figure 1. **The relationship between target and source pixels in position**

2.2. Adaptive Newton interpolation algorithm

The same as the third-order Newton algorithm mentioned in section 2.1, bilinear and bicubic interpolation methods don't estimate the relativity among the source pixels when computing the target pixel. As a result, these methods lead to edge blurring and a loss in the image details containing abundant information. Because the relativity of pixels conceals in the image, we can select an appropriate parameter to show the relativity.

If equation (4) is used for interpolation, every target pixel will be attained according to the three adjacent source pixels. In figure 1, the gray-value of the pixel N is computed by either f_0, f_1, f_2 or f_1, f_2, f_3 with equation (4). Assuming f_0, f_1, f_2, f_3 are located at the uniform area of an image, i.e., all pixel gray values differ little, the interpolation results are almost equal by either group of the two; while if f_0, f_1, f_2, f_3 are located at the detail and edge area, the results of computing the two groups of source pixels with equation (4) are much different. How to select the source pixel group is the key problem to solve in this paper.

According to equation (2), the second-order differences of both f_0, f_1, f_2 and f_1, f_2, f_3 are defined by equation (6) and equation (7), respectively.

$$\Delta^2 f_0 = f_2 - 2f_1 + f_0 \qquad (6).$$

$$\Delta^2 f_1 = f_3 - 2f_2 + f_1 \qquad (7).$$

Where, the absolute value of the second-order difference weighs the relativity of three consecutive pixels. The smaller is the absolute value, the bigger is the relativity, vice versa. In figure 1, it is more reasonable to compute the gray value of target pixel N by using the source pixels with bigger relativity than smaller relativity. The bigger is relativity of the consecutive pixels, the more is possibility of pixels being in the same area of an image.

As mentioned above, either of the two groups is automatically selected to interpolate by comparing the absolute value of the second-order difference of the two groups of source pixels. Thus, improved function to compute the gray value of the target pixels N is as follows,

$$N_2 = \begin{cases} f_0 + \Delta f_0 t + \frac{\Delta^2 f_0}{2!} t(t-1) & \text{if} \quad |\Delta^2 f_0| \le |\Delta^2 f_1| \\ f_1 + \Delta f_1 (t-1) + \frac{\Delta^2 f_1}{2!} (t-1)(t-2) & \text{if} \quad |\Delta^2 f_0| > |\Delta^2 f_1| \end{cases} \qquad (8).$$

Here, f_0, f_1 are the source pixels shown in figure 1, t is the distance between the target pixel N and the source pixel f_0. The first-order difference Δf_0, Δf_1 are computed by function (1), and the second-order $\Delta^2 f_0$, $\Delta^2 f_1$ are computed by function (6) and function (7). By comparing $|\Delta^2 f_0|$ with $|\Delta^2 f_1|$, a group of appropriate source pixels are selected to compute the gray value of the target pixel N_2.

3. Implementation of the algorithm

The adaptive Newton interpolation algorithm denoted by function (8) has lower complexity than the third-order non-adaptive Newton interpolation or bicubic interpolation. To compute the gray value of a target pixel needs five times multiplication and nine times addition by the third-order Newton interpolation, or five times multiplication and ten times addition by bicubic interpolation. But it only needs three times multiplication and five times addition to implement interpolation for one target pixel by our proposed algorithm. Compared with the two former methods, the computational complexity of our proposed algorithm reduces about 40%.

Figure 2 illustrates the signal flow of implementing the proposed algorithm by FPGA in real-time conditions. The f_0, f_1, f_2, f_3 are provided by tandem memory FIFO (first-in first-out) or random memory. The first-order difference values Δf_0, Δf_1, Δf_2 are computed by parallel addition operation, and the second-order difference values $\Delta^2 f_0$ and $\Delta^2 f_1$ are acquired by the same operation. The comparator outputs a logical value by comparing $\Delta^2 f_0$ with $\Delta^2 f_1$, and the logical value acts as a judging condition for three multiplexers. A set of values selected by the three multiplexers input the module "Σ" (see figure 2), then the interpolation result N_2 is computed.

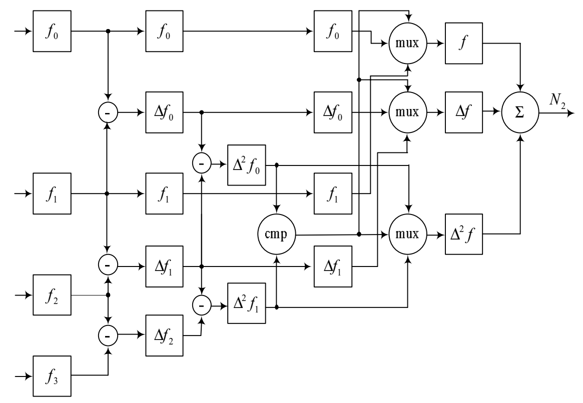

Figure 2. **The diagram of signal flow of our proposed algorithm's implementation**

4. Experimental results

Implementing our proposed algorithm and many other classical interpolation methods in Matlab, the subjective visual effect and PSNR of all methods are compared. Our proposed algorithm is described in Verilog HDL according to the diagram in figure 2. Then the source codes are synthesized into object codes and the system function is verified in evaluation board by downloading object codes in FPGA.

4.1. Simulation and evaluation of algorithms

As mentioned in section 2, most of the interpolation methods can achieve almost identical visual effect in uniform areas (see area A in figure 3(a)) after processing of image resizing. But in detail (see area C in figure 3(a)) and edge areas (see area B in figure 3(a)), visual effect is much different due to the great gray gradient of adjacent pixels. Therefore, we compare the visual effect of image resizing in detail and edge areas.

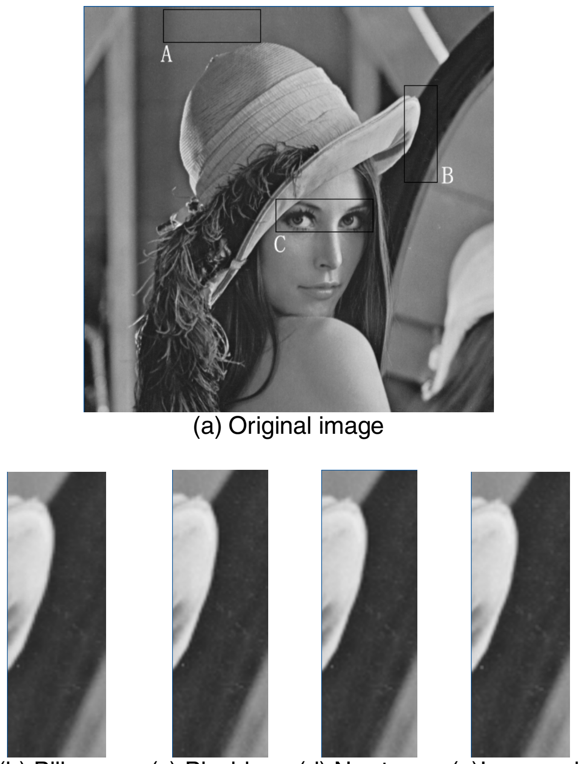

Figure 3. **Visual effect of resizing image by different interpolation algorithm**

Figure 3(a) is original Lena image, the size of which is 512×512, and figure 3(b~e) are the enlarged images corresponding to area B of Lena's hat edge in original image by different interpolation algorithms. The size of area B is 40×120 in figure 3(a). In figure 3(b~e), the size of images are all 160×480, and corresponding interpolation methods are bilinear, bicubic, the third-order Newton and our proposed improved Newton interpolation in turn. In figure 3(b), jagged edge appears in hat edge area, and the whole effect is blurring. Comparing figure 3(c) with figure 3(b), the whole visual effect improves. But "ringing" appears near hat edge and the contour is indistinct. Figure 3(d) and figure 3(c) are both the results of cubic interpolation, so their differences in visual effect are unconspicuous. In figure 3(e), the interpolation result by our proposed algorithm has a quite clear contour and pleasant visual effect.

PSNR [5] are computed to evaluate every algorithm applied to resize area A, B, C and the complete Lena image in figure 3(a).

Table 1. **Comparison of the PSNR (dB)**

Images	Bilinear	BiCubic	Newton	Improved
Area A	48.1809	47.9875	47.8336	49.6592
Area B	45.8194	48.0635	47.6040	49.1226
Area C	45.8670	48.0688	47.6776	49.1740
Lena	46.2818	48.0086	47.6126	49.4249

In figure 3(a), area A, B and C are uniform area, edge area and detail area in turn. Both area A and area C are processed as follows: First, images are enlarged to 160×480 from the original size 40×120; Second, the enlarged images are reduced to 40×120 from 160×480. Similar process is carried out to area B, its scaling sequence is $120\times40\rightarrow480\times160\rightarrow120\times40$. And the scaling sequence of the complete Lena image is $512\times512\rightarrow2048\times2048\rightarrow512\times512$. Table 1 shows that the differences of all PSNR values are tiny when using different algorithms to resize the uniform area, but the PSNR values of our proposed algorithm are greater than those of others. Although, bicubic and the third-order Newton method excel bilinear interpolation in resizing detail area C, edge area B and the complete Lena image, the PSNR values of our proposed adaptive quadratic Newton interpolation is evidently greater those of others.

4.2. Verified by FPGA

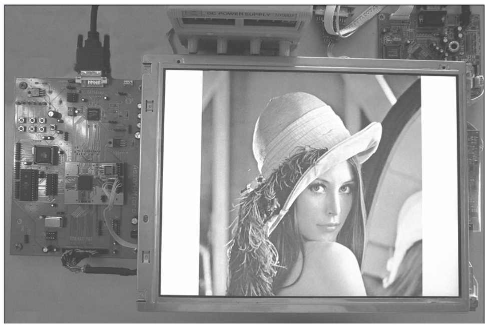

Figure 4. **Implementation of image resizing algorithms in FPGA**

All interpolation algorithms mentioned in section 4.1 are programmed in Verilog HDL. These codes are synthetized, respectively, and the object codes are downloaded in FPGA on the evaluation board illustrated by Figure 4. Image and timing signals with different resolutions provided by DVI are converted into RGB and corresponding timing control signals. Controlled by the same timing signals, the three components of RGB are respectively resized in horizontal and vertical directions, then the processed images are displayed on LCD panel with resolution of 1024×768. Visual effects of images resized by different algorithms are entirely consistent with the simulation results in section 4.1.

5. Conclusion

In this paper, according to the relativity judged by the second-order difference of the adjacent pixel gray values, an adaptive image resizing algorithm based on Newton interpolation function is presented. Once the relativity is judged, every order difference value is provided to the second-order Newton interpolation function at the same time. So computational complexity of our proposed algorithm reduces to about 60% of that of the bicubic interpolation, and the corresponding circuit architecture is very simple. Experimental results show that the visual effect of our proposed method excels that of bicubic interpolation in resizing images, and the PSNR values of the resized image by our proposed algorithm are evidently greater than those of other classical interpolation algorithms. In conclusion, our proposed algorithm implements image interpolation in high efficiency, and it is especially well applied to real-time image resizing.

6. References

[1] P. Thévenaz, T. Blu, and M. Unser, "Interpolation revisited", *IEEE Trans. Medical Imaging*, the Institute of Electrical and Electronics Engineers, Inc., U.S.A, July 2000, pp. 739 - 758.
[2] C. L. Huang, and K. C. Chen, "Direction moving averaging interpolation for texture mapping", *Graphical Models and Image Processing*, Academic Press, U.S.A, July 1996, pp. 301-313.
[3] M. Unser, A. Aldroubi, and M. Eden, "Enlargement or reduction of digital images with minimum loss of information", *IEEE Trans. Image Processing*, the Institute of Electrical and Electronics Engineers, Inc., U.S.A, March 1995, pp. 247-258.
[4] K. Jensen, and D. Anastassiou, "Subpixel edge localization and the interpolation of still images", *IEEE Trans. Image Processing*, the Institute of Electrical and Electronics Engineers, Inc., U.S.A, March 1995, pp. 285-295.
[5] Y. Lin, H. H. Chen, and Z. H. Jiang, etc., "Image Resizing with Raised Cosine Pulses", *Proceedings of 2004 International Symposium on Intelligent Signal Processing and Communication Systems*, the Institute of Electrical and Electronics Engineers, Inc., U.S.A, November 2004, pp. 581-585.