Fast and Effective Interpolation Using Median Filter

Jian Zhang[1,*], Siwei Ma[2], Yongbing Zhang[1], and Debin Zhao[1]

[1] Department of Computer Science, Harbin Institute of Technology, Harbin 150001, P.R. China
[2] School of Electronics Engineering and Computer Science, Peking University, Beijing 100080, P.R. China
{zhangjian,swma,ybzhang,dbzhao}@jdl.ac.cn

Abstract. This paper proposes a fast and effective image interpolation algorithm using median filter. The interpolation algorithm is composed of two steps. First, a non-linear iterative procedure is utilized to interpolate the pixel whose direction can be easily determined by local information. Second, according to the introduced assumption that image interpolation can be regarded as a local image filtering process, the remaining pixels are interpolated by the proposed fast median filter method. Experimental results show that the proposed algorithm provides better performance than traditional techniques (e.g. bilinear interpolation, bicubic interpolation) both in subjective quality and objective quality with similar complexity. In particular, the proposed method is comparable to the well-known NEDI algorithm in visual quality, however, with much lower computational complexity. Therefore, the proposed algorithm can be exploited for real-time applications due to the merits of low computational complexity and good image quality.

Keywords: Image interpolation, median filter, image zooming.

1 Introduction

Image interpolation aims to reconstruct a high-resolution (HR) image from the associated low-resolution (LR) image, which is often necessary for a variety of applications, such as medical imaging, remote sensing, SDTV-To-HDTV conversion, etc.

In practice, two of the most commonly used techniques are bilinear interpolation and bicubic interpolation due to their simplicity and fast implementation. Although these two methods (especially bicubic interpolation) seem to keep relatively high PSNR, blurred and jagged effects are often observed because of the lack of adaptability, which degrades the visual quality of the interpoated image.

Many methods [1, 2, 3, 4, 5] have been recently proposed to acquire high quality zooming images. However, the common disadvantage of these methods is higher computational complexity, which limits the real-time application. Among these methods, the well known covariance-based adaptive interpolation algorithm NEDI (New

* This work was supported in part by National Science Foundation of China (60803068), and National Basic Research Program of China (973 Program, 2009CB320904).

P. Muneesawang et al. (Eds.): PCM 2009, LNCS 5879, pp. 1176–1186, 2009.

Edge Directed Interpolation) [1] seems to provide the best visual quality, whereas its principal drawback is the prohibitive computational complexity. To speed up the interpolation process, some fast locally adaptive interpolation methods [6, 7] are proposed to produce better visual quality images, but they still can not provide images visually competitive with NEDI. Accordingly, it is hard to achieve an ideal balance between quality and computational complexity.

This paper proposes a novel approach to make a good trade-off between image quality and computational complexity. We first introduce a basic assumption that image interpolation can be considered as a local image filtering process, which gives a new interpretation of image interpolation. Next, a non-linear iterative procedure and a median-filter-based method are fused to generate our proposed algorithm. The non-linear iterative procedure is utilized to interpolate the pixel whose direction can be easily determined by the properties of its neighboring pixels. The median-filter-based method is employed to interpolate the remaining pixels. At last, a simple and fast method to calculate the median value of four neighboring pixels is designed to speed up the median filter process. Experimental results show that our proposed method provides better performance than traditional techniques (e.g. bilinear interpolation, bicubic interpolation) both in subjective quality and objective quality with similar complexity. Moreover, the subjective quality of our algorithm is comparable to the well-known NEDI algorithm with much lower computational complexity, which strongly proves that our assumption is reasonable and our algorithm is effective.

The rest of this paper is organized as follows. Section 2 introduces our basic assumption. Section 3 presents the details of our proposed fast and effective interpolation algorithm. Experimental results are reported in Section 4. Section 5 concludes this paper.

2 Basic Assumption

Our basic assumption is that image interpolation can be regarded as a local image filtering process. Therefore, as viewed from image filtering, we can consider interpolated pixels as noisy pixels and then give a new interpretation of image interpolation as follows:

Referring to Fig. 1,

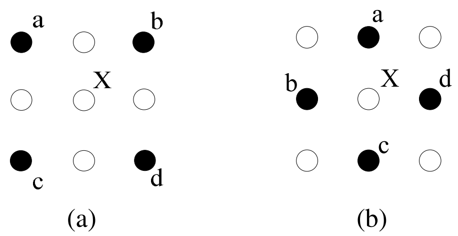

Fig. 1. The picture illustrates the basic assumption

1) if X=(a+b+c+d)/4, then X can be seen as a noisy pixel which makes a mean filter;
2) if X=median([a b c d]), then X can be seen as a noisy pixel which makes a median filter.

Compared with traditional image filtering, the only difference is that there are four rather than nine pixels available around the noisy pixel in a 3×3 template. Thus by associating image interpolation with image filtering our assumption is reasonable.

In this paper, our contribution is to design a two-step interpolation process: first, we interpolate the pixel whose direction can be easily determined via a simple and effective locally adaptive method, and second, we estimate the remaining pixels utilizing another fast and effective method. But the challenge is how to choose and design these two methods? The answer is as follows. As to the first step, the non-linear iterative method proposed in the locally adaptive zooming algorithm (LAZA) [6] is directly adopted due to its simplicity and effectiveness. As to the second step, one straightforward method is to estimate the pixel using the mean of its neighboring pixels. However, from the point view of our assumption, it can be considered as a local mean filtering process. This is owing to the following two significant reasons: a) in image interpolation the overwhelming majority of those pixels whose directions are not easily determined are edge pixels, which contain plenty of edge information; b) according to image filtering theory [8], the median filter is superior to the mean filter in the property of maintaining edge information. Consequently, it is believed that when exploiting median filter instead of mean filter in the second step we can get better images in terms of both objective and subjective quality.

3 Proposed Fast and Effective Algorithm

In this section, we describe the details of our proposed algorithm. For the sake of simplicity the zooming factor used here is two in both horizontal and vertical directions. As mentioned before, the proposed algorithm consists of two steps. The first step employs the non-linear iterative procedure to interpolate the pixel whose direction is easily determined. The second step adopts median filter method to interpolate the remaining pixels by virtue of its superior property of preserving edges and suppressing jagged and other visual effects. A fast median filter algorithm for four numbers is also presented later for keeping fast speed. Hence, with low complexity the proposed interpolation algorithm obtains much better images both in objective and subjective quality. In a word, two intriguing merits of our proposed algorithm are simplicity and effectiveness, which will be fully verified by the following experiments.

3.1 First Step: Non-linear Iterative Procedure

The non-linear iterative procedure also comprises three stages, as described in the following.

3.1.1 Image Expansion

This stage is to expand the LR image $I_L(N\times M)$ into a HR image $I_H(2N\times 2M)$, subject to $I_L(i,j) = I_H(2i-1,2j-1)$, $1\le i\le N, 1\le j\le M$. Referring to Fig. 2, the interpolation problem is to estimate the white dots in HR image I_H.

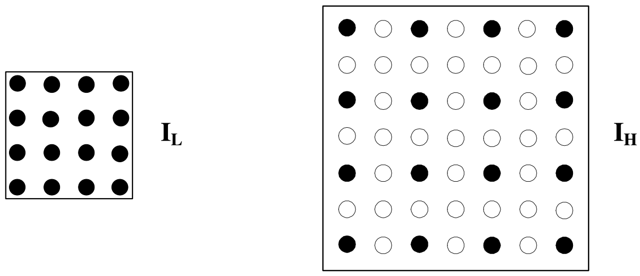

Fig. 2. The picture shows the first stage of expansion

3.1.2 Interpolation of Pixels I_H (2i,2j)

At the second stage the algorithm deals with the pixels in I_H, whose coordinates are both even (e.g. the pixels labeled with X in Fig. 3a). For reference, we will use the letters a, b, c, d, H1, H2, V1 and V2, as in Fig. 3a, to denote the original LR image pixels and the HR image pixels surrounding the pixel X. Through simple calculation of these four pixels, the pixel X whose direction is easy to judge can be directly interpolated along its direction.

The process is as follows:

Smooth region: if |range(a,b,c,d)|<T1, then X = (a+b+c+d)>>2
SW-NE direction: if |a-d|>T2 && |a-d|>>|b-c|, then X = (b+c)>>1
NW-SE direction: if |b-c|>T2 && |b-c|>>|a-d|, then X = (a+d)>>1
NS direction: if |a-d|>T1 && |b-c|>T1 && (a-d)×(b-c)>0, then
H1 = (a+b)>>1; H2 = (c+d)>>1; X is left undefined
EW direction: if |a-d|>T1 && |b-c|>T1 && (a-d)×(b-c)<0, then
V1 = (a+c)>>1; V2 = (b+d)>>1; X is left undefined

where range(a,b,c,d) = max{a,b,c,d}-min{a,b,c,d}; >> represents right shift operation; |Y| denotes absolute value of number Y; T1 and T2 denote suitable thresholds.

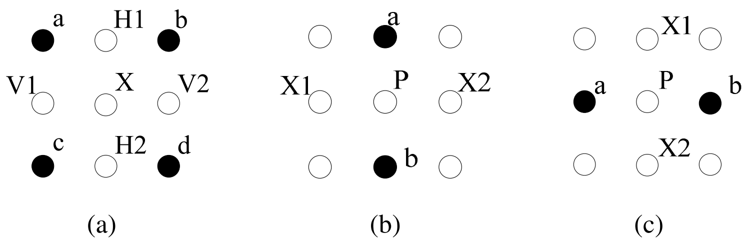

Fig. 3. The picture of pixels to be interpolated

After this stage there will be many pixels left undefined, which will be handled in the following stages.

3.1.3 Interpolation of Pixels I_H (2i-1,2j) and I_H (2i,2j-1)

This stage will deal with pixels with at least one odd coordinate of I_H, as denoted by letter P in Fig. 3b and Fig. 3c. The two cases illustrated in Fig. 3b and Fig. 3c will be treated similarly in the following discussion. Referring Fig. 3b (or Fig. 3c), we use the letters a and b to denote the pixels above and below (or left and right) of pixel P, and use the letters X1 and X2 to denote the pixels left and right (or above and below) of pixel P. It is noted that a and b have already been assigned a value respectively as original LR image pixels. Then according to whether the values of pixels X1 and X2 have been assigned in the previous stage, there are two situations as follows:

```
if X1 or X2 are undefined, do
   if |a-b|<T1, P = (a+b)>>1
else if
   X1X2 direction: if |a-b|>T2 && |a-b|>>|X1-X2| then P = (X1+X2)>>1
      ab   direction: if |X1-X2|>T2 && |X1-X2|>>|a-b| then P = (a+b)>>1
else
   P is left undefined
```

where >> represents right shift operation; |Y| denotes absolute value of number Y.

Accordingly, through the first step, every pixel whose direction is easy to determine by its neighboring pixels has been assigned a value. Next, in the second step, the remaining undefined pixels will be eventually interpolated.

3.2 Second Step: Interpolation Using Median Filter

From the previous analysis, it is known that the majority of the remaining undefined pixels are edge pixels including a large number of edge information. Because edge is important in image interpolation, these undefined pixels play a crucial role for image quality both in objective and subjective. According to our assumption that median filter is superior to mean filter for preserving edge information, we adopt median-filter-based interpolation method .

Referring to Fig. 3a, for every pixel X with both even coordinates and an undefined value, the median value of the pixels a, b, c ,d in the neighbor is assigned to X, i.e. X=median(a,b,c,d). Note that a, b, c, d come from the original LR image that are not affected by the smoothing operations done in the previous three stages, which is key to preserve visual sharpness. Afterwards, observe that at this time X1 and X2 have been estimated before, and a and b are original LR image pixels, as shown in Fig. 3b and 3c. Hence, for every pixel P with at least one odd coordinate and an undefined value, we assign P the median value of the pixels a, b, X1, X2 surrounding it, i.e. P=median(a,b,X1,X2). We choose only four pixels neighboring the interpolated pixel because it not only has fast speed, but also can effectively retain the edge information and reduce the affect of the first three stages, which is just what we need.

3.3 Fast Median Method

In order to speed up the median filter process, we design a fast method to calculate the median value of four pixels. The method enables our whole algorithm to keep the fast speed comparable with Bicubic interpolation method. Furthermore, its complexity is just one-sixth of the MATLAB's own median algorithm in MATLAB environment.

Suppose A is a four-dimension column vector, i.e. A = [a1 a2 a3 a4], then the fast algorithm is

$$\text{median}(A) = (\text{SUM}(A)-\text{MIN}(A)-\text{MAX}(A))>>1$$

where >> represents right shift operation; SUM(A) = a1+a2+a3+a4; MIN(A) = min{a1,a2,a3,a4}; MAX(A) = max{a1,a2,a3,a4}.

4 Experimental Results

In this section, We first compare the objective and subjective qualities of different interpolation algorithms including Bilinear, Bicubic, NEDI [1], LAZA [6] and Ours. Ours is short for our proposed algorithm. Furthermore, in order to verify the simplicity of our algorithm, the comparison of time computational complexity of Bicubic, NEDI, LAZA and Ours in MATLAB are also given. In this paper, the thresholds T1 and T2 are set to be 4 and 3 respectively, and the algorithm LAZA [6] is implemented with the parameter q = 1.

4.1 Objective Quality Comparison

In this subsection, eight grayscale images are tested, including Lena (510×510), Girl (512×512), Boat (512×512), Children (492×324), Cap (768×512), Peppers (512×512), Sea (768×512), Airplane (768×512). The quality measure in our experiments is the classical Peak Signal to Noise Ratio (PSNR). We first downsample directly the HR images to get the corresponding LR images, from which the interpolated HR images are reconstructed by the proposed and competing methods. By comparing the original images with the interpolated images, we can calculate the PSNR. The PSNR comparison is shown on Table 1. From Table 1, it is obvious to see that our proposed interpolation algorithm achieves the highest average PSNR.

Table 1. PSNR Comparison(Uint: dB)

Picture	Lena	Girl	Boat	Children	Cap	Peppers	Sea	Airplane	Average
Bilinear	33.41	31.8	29.15	30.2	33.37	31.48	30.29	30.12	**31.23**
Bicubic	33.98	31.54	29.27	30.45	33.45	31.59	30.05	30.17	**31.31**
NEDI	33.7	31.95	29.08	30.69	33.4	29.32	30.08	28.74	**30.87**
LAZA	33.38	31.85	29.17	30.38	33.33	31.53	30.27	30.18	**31.26**
Ours	33.62	31.94	29.27	30.78	33.5	31.74	30.24	30.49	**31.45**

4.2 Subjective Quality Comparison

In order to acquire a comprehensive subjective quality comparison, two types of methods are employed. One is directly to enlarge the original small image by different interpolation algorithms, and the other is to interpolate the LR image which is obtained by directly downsampling the corresponding HR image.

In this subsection, four examples of visual quality of the test interpolation methods are shown. Figs.4-5 are generated by the former method, and Figs. 6-7 are generated by the latter. It is obvious to observe that our proposed method obtains similar subjective quality compared with that of NEDI and the performance superior to the rest of the three methods by suppressing many of the annoying jagged and other visual artifacts. Furthermore, as to Lena, it is noted that like NEDI, our proposed method acquires much better subjective quality than Bicubic method by dramatically suppressing the jagged effect in face and shoulder, although its PSNR is slightly lower than Bicubic's, which fully verifies the advantage of our algorithm in subjective quality.

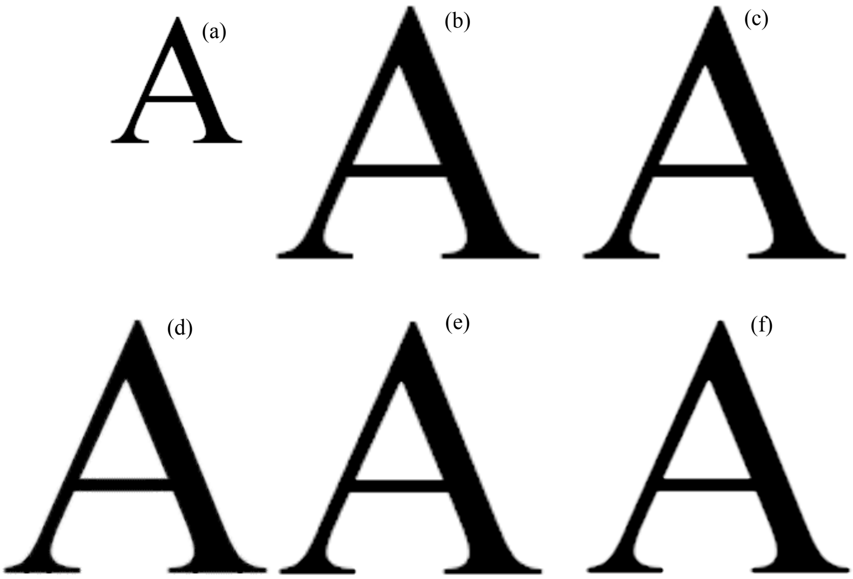

Fig. 4. (a) Original; (b) Bilinear; (c) Bicubic; (d) NEDI [1]; (e) LAZA [6]; (f) Ours

4.3 Time Computational Complexity Comparison

A comparison of time computational complexity of Bicubic, NEDI, LAZA and Ours in the environment of MATLAB are given in Table 2. The situation of our

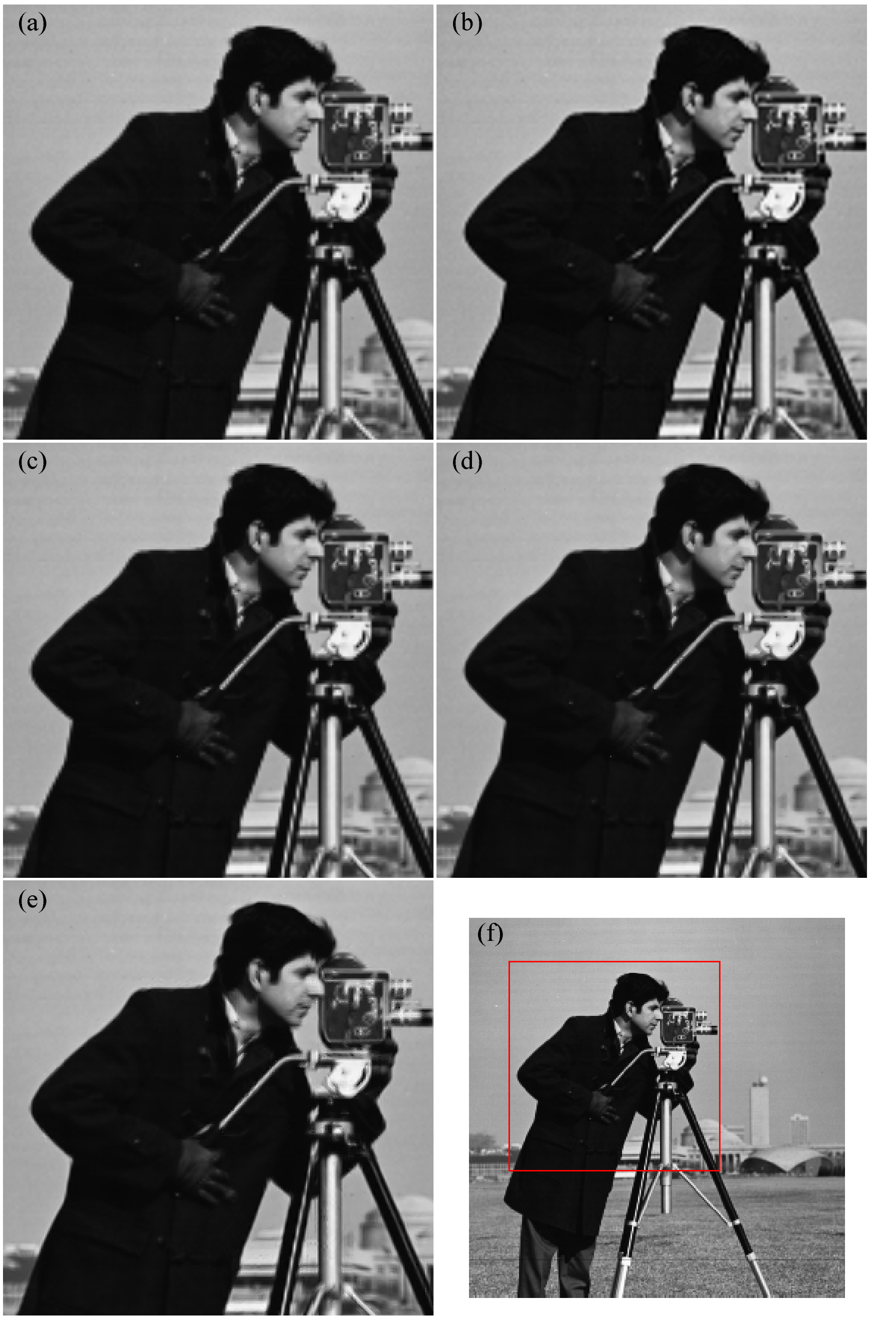

Fig. 5. (a) Bilinear; (b) Bicubic; (c) LAZA [6]; (d) Ours; (e) NEDI [1]; (f) Original

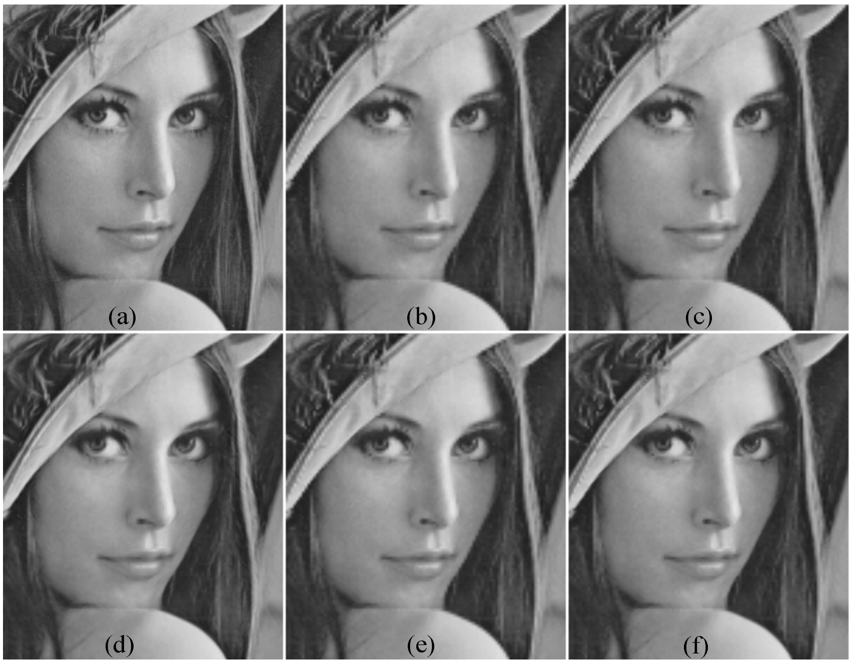

Fig. 6. (a) Original; (b) Bilinear; (c) Bicubic; (d) NEDI [1]; (e) LAZA [6]; (f) Ours

experiment is as follows, MATLAB 7.3, Intel Core 2 CPU with 1.83 GHz and 1 GB memory.

From Table 2, it is concluded that a) the time complexity of Ours is only an increase by 35% compared with that of LAZA; b) our proposed algorithm is merely twice the time complexity of Bicubic; and c) Ours is only one-thirtieth of the time complexity of NEDI. As a result, our proposed algorithm still maintains the fast property, which is quite demanding for real-time applications.

Table 2. Time Computational Complexity Comparison in MATLAB(Unit: second)

Picture	Lena	Girl	Boat	Children	Cap	Peppers	Sea	Airplane	Average
Bicubic	0.58	0.6	0.6	0.37	0.91	0.61	0.9	0.85	**0.68**
NEDI	32.51	49.22	34.55	19.77	55.24	33.2	49.29	43.79	**39.7**
LAZA	0.83	0.91	0.92	0.55	1.22	0.88	1.32	1.07	**0.96**
Ours	1.13	1.19	1.2	0.71	1.68	1.16	1.77	1.56	**1.3**

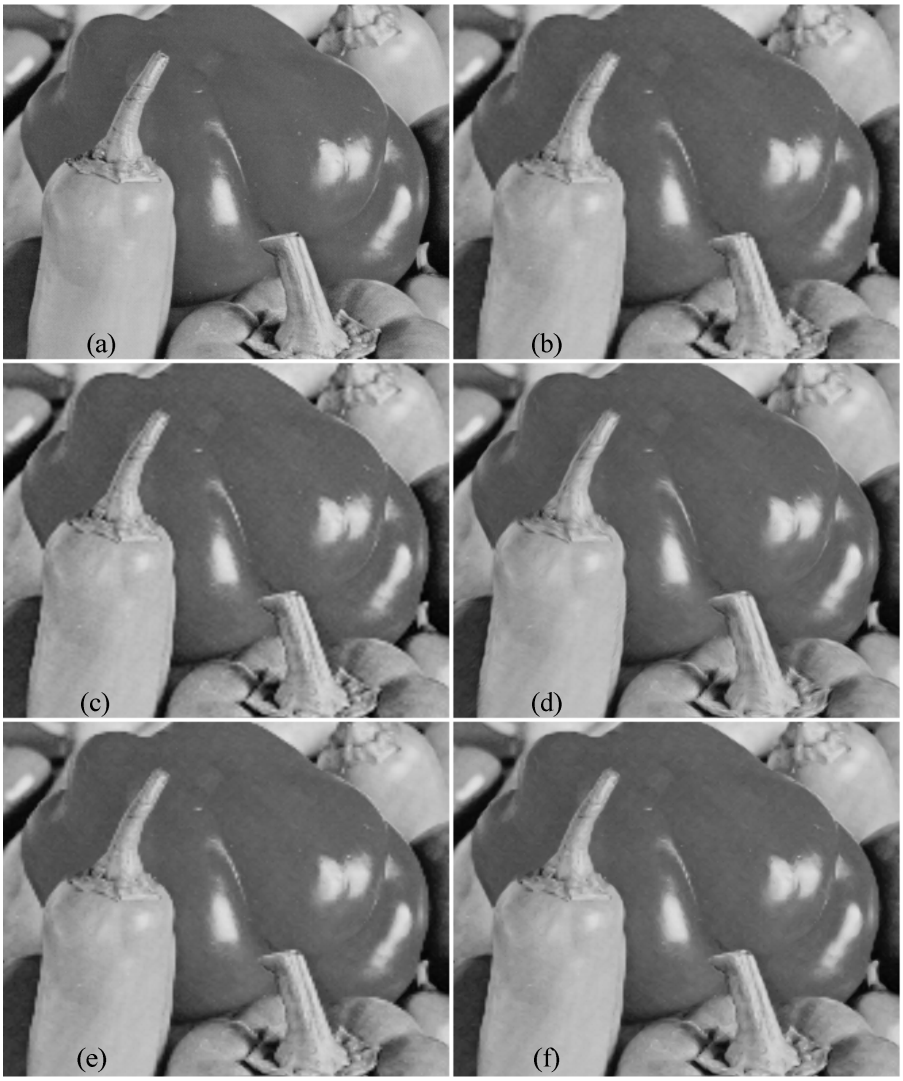

Fig. 7. (a) Original; (b) Bilinear; (c) Bicubic; (d) NEDI [1]; (e) LAZA [6]; (f) Ours

5 Conclusions

In this paper, we presented a novel approach to make a good trade-off between image quality and computational complexity. Firstly, a basic assumption that image interpolation can be considered as a local image filtering process was made. Then, according to this assumption, a fast and effective interpolation algorithm was proposed via the combination of a non-linear iterative procedure and a median-filter-based method. The non-linear iterative procedure was utilized to interpolate the pixel whose direction can be easily determined by its neighboring pixels, while the median-filter-based method was employed to interpolate the remaining pixels. Finally, to reduce the complexity of processing median filter, we introduced a simple and fast method to calculate the median value of four neighboring pixels. Experimental results showed that the proposed algorithm not only quickly acquired better interpolation images in visual quality than traditional techniques, but also surpassed traditional techniques in objective quality (PSNR), which demonstrated that the proposed algorithm had the merits of simplicity and effectiveness. In the future work, we will apply the presented algorithm to video coding by virtue of low computational complexity and good image quality.

References

1. Li, X., Orchard, M.T.: New edge-directed interpolation. IEEE Trans. on Image Processing 10(10), 1521–1527 (2001)
2. Wang, Q., Ward, R.K.: A New Orientation-Adaptive Interpolation Method. IEEE Trans. on Image Processing 16(4), 889–900 (2007)
3. Zhang, L., Wu, X.: An Edge-Guided Image Interpolation Algorithm via Directional Filtering and Data Fusion. IEEE Trans. on Image Processing 15(8), 2226–2238 (2006)
4. Yang, S., Kim, Y., Jeong, J.: Fine edge-preserving technique for display devices. IEEE Trans. on Consumer Electronics 54(4), 1761–1769 (2008)
5. Cha, Y., Kim, S.: The Error-Amended Sharp Edge (EASE) Scheme for Image Zooming. IEEE Trans. on Image Processing 16(6), 1496–1505 (2007)
6. Battiato, S., Gallo, G., Stanco, F.: A locally adaptive zooming algorithm for digital images. Image and Vision Computing 20(11), 805–812 (2002)
7. Chen, M., Huang, C., Lee, W.: A fast edge-oriented algorithm for image interpolation. Image and Vision Computing 23(9), 791–798 (2005)
8. Pitas, I., Venetsanopoulos, A.N.: Nonlinear Digital Filters: Principles and Applications. Kluwer, Norwell (1990)