An Edge Based Adaptive Interpolation Algorithm for Image Scaling

Wanli Chen, Hongjian Shi
Department of Electrical and Electronic Engineering
Southern University of Science and Technology, Shenzhen, Guangdong, China
chenwl@mail.sustc.edu.cn, shihj@mail.sustc.edu.cn

***Abstract*—In this paper, the Canny and Sobel operators are combined to scale an image and to produce an image with high resolution and clear edges. The Canny operator is used at first to detect edges of the objects inside the original image. After that, four Sobel operators in different directions are applied to detect the edge directions. The direction of the edge at one edge point is determined by comparing the first derivatives of the intensity at edge pixels. The image interpolation is carried out adaptively according to the determined edge direction. For the homogeneous areas, the bilinear interpolation is applied. After image scaling, one method to suppress the zig-zag noises is applied to enhance the output image. The first and second steps of such zig-zag suppression are the same with the proposed image scaling algorithm. Then the pixels around the edge pixels are specially modified. The experimental results show that our proposed algorithm could produce scaled images with high resolution and well-preserved edges.**

Keywords—Image scaling; edge detection; Sobel operator; Canny operator; bilinear interpolation; bicubic interpolation

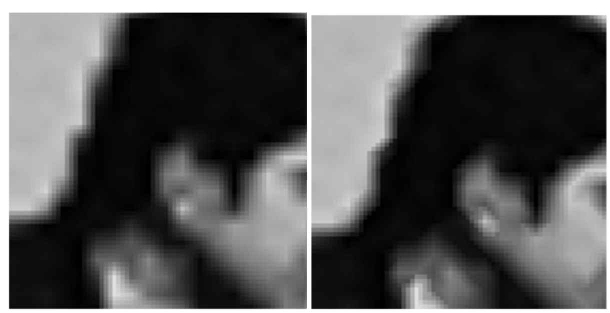
I. Introduction

Image scaling is important in our life, this technology has already been used in our daily life [1]-[2]. Three traditional interpolation algorithms are commonly used in image scaling. The nearest neighbor interpolation [3] is the fastest algorithm. However, the produced images are the worst. The bilinear interpolation [3] performs well in smooth area of the image but it usually produces blurred edges. The image produced by the bicubic interpolation [4] has the best quality among the three mentioned methods, but it has the highest computational complexity and the produced images do not always look natural.

The edge adaptive interpolation is a popular image scaling algorithm because it could produce crispy images with high resolution. This kind of interpolation usually detects image edges or heterogeneous areas at first, then interpolates these areas using different algorithms. The homogeneous areas are interpolated usually by the bilinear interpolation, because it performs well in smooth regions. Many edge adaptive algorithms are developed in recent years. Shi and Ward [5] proposed a Canny edge based interpolation algorithm, the results showed that it could yield some light zig-zag artifacts. Lai et al. [6] proposed an edge adaptive algorithm by using their own edge detector and the produced images have sharper edges than those that the bilinear and bicubic interpolations produce. Jiang et al. [7] proposed a Prewitt operator based algorithm that achieved lower computational complexity and better edge reservation than using the bilinear interpolation. Chen and Shi [8] developed a fast image scaling algorithm based on four Sobel operators and the results showed that the algorithm performs better than the bilinear and bicubic algorithm and the produced images have clear and smooth edges. Our goal here is to search for a better edge based adaptive interpolation method to scale images and to produce images with higher quality.

II. Related Work

A fast edge adaptive interpolation method was proposed in [7], the authors improved the bilinear interpolation method by optimizing the interpolation method in the edge areas. There four directional Prewitt operators are applied to find the edges. When the edges are detected, the edges will be interpolated linearly by simply using two neighbor pixels. The improved version of [7] is proposed in our previous work [8]. Four directional Sobel operators were used in [8], which had better noise suppression performance than the Prewitt operator. However, the scaled image still has obvious zig-zag edges (see Fig. 1). Thus, a method for zig-zag suppression should be applied to enhance the edge performance. The paper [5] introduced a simple but effective method for zig-zag suppression, which was to use the neighbor pixels of the edge pixels. The improved method of [8] is proposed in this paper by using the Canny operator and the optimized method of [5] is also be applied in our new algorithm.



(a) Bilinear (b) Algorithm in [5]

Fig. 1. Zig-zag performance comparison.

In our new proposed algorithm, the image interpolation algorithm in [8] is improved. The Canny operator is used for original image at first and four direction Sobel operators are applied just for detected edge pixels, which will improve the output images and lower the computational complexity. Then an optimized algorithm in [5] is applied by adding more interpolation directions. Furthermore, the pixels around the edge pixels are modified based on different orientations. The proposed algorithm performs better than our recent work [8] and Shi and Ward [5]. The proposed algorithm performs better than the bilinear and bicubic interpolations when the processed image has many edges and textures.

D1: 0°

1	2	1
0	0	0
-1	-2	-1

D2: 90°

1	0	-1
2	0	-2
1	0	-1

D3: 45°

2	1	0
1	0	-1
0	-1	-2

D4: 135°

0	-1	-2
1	0	-1
2	1	0

Fig. 2. Four proposed Sobel operators.

III. PROPOSED ALGORITHM

In our proposed algorithm, the four directional Sobel operators are used as the algorithm in [8] (Fig. 2), but the operators are not used to every pixel of the original image. The proposed algorithm is composed of three parts: the Canny Edge detection, the Sobel Edge-Adaptive Interpolation, and the Canny-Sobel zig-zag Suppression.

A. *Canny Edge Detection*

The Canny operator is used at first to detect the edge of the original image. The Canny operator performs better than the Sobel operators for three reasons [5]:

1) It could separate edge points and non-edge points precisely.

2) The detected edge points are usually at the center of edge.

3) The detected edges are usually one pixel width.

Thus, the Canny operator is a strong candidate for image edge detection. After the Canny edge detection, the Sobel operators are used to determine the edge directions.

B. *Sobel Edge-Adaptive Interpolation*

1) Determining Adaptive Thresholds

The threshold values T_x , T_y , T_{45} and T_{135} denote the thresholds for horizontal, vertical and diagonal directions respectively. They are important for edge direction determination and are used as follow (here arctan(l) denotes the phase of the edge l at current point):

If $\left|\frac{\partial f}{\partial x}\right| \geq \left|\frac{\partial f}{\partial y}\right|$, and $\left|\frac{\partial f}{\partial x}\right| \geq T_x$, the edge direction at this point is considered in x direction.

If $\left|\frac{\partial f}{\partial y}\right| > \left|\frac{\partial f}{\partial x}\right|$, and $\left|\frac{\partial f}{\partial y}\right| \geq T_y$, the edge direction at this point is considered in y direction.

If $\left|\left|\frac{\partial f}{\partial l}\right|_{\arctan(l)=45°}\right| \geq \left|\left|\frac{\partial f}{\partial l}\right|_{\arctan(l)=135°}\right|$, and $\left|\left|\frac{\partial f}{\partial l}\right|_{\arctan(l)=45°}\right| \geq T_{45}$, the edge direction at this point is considered in 45° diagonal direction.

If $\left|\left|\frac{\partial f}{\partial l}\right|_{\arctan(l)=135°}\right| \geq \left|\left|\frac{\partial f}{\partial l}\right|_{\arctan(l)=45°}\right|$, and $\left|\left|\frac{\partial f}{\partial l}\right|_{\arctan(l)=135°}\right| \geq T_{135}$, the edge direction at this point is considered in 135° diagonal direction.

To get the adaptive threshold values, the Sobel operators are used on the Canny detected edge at first. Then the results are stored in four arrays A_x , A_y , A_{45} , and A_{135}. The gradient $\left|\frac{\partial f}{\partial y}\right|$ is defined as

$$\left|\frac{1}{4}[f(x-1,y-1)-f(x-1,y+1)]+\frac{1}{2}[f(x,y-1)-f(x,y+1)]+\frac{1}{4}[f(x+1,y-1)-f(x+1,y+1)]\right|$$

through the Sobel operator. Thus, the adaptive threshold value could be defined as:

$$T_y = \frac{\mu + \sigma}{4}$$

Where μ is the average value of A_y and σ is the standard deviation of A_y. T_x , T_{45} and T_{135} are defined similarly.

2) Determining Edge Direction

After determination of the threshold values, the Sobel operators are used again to find direction of every edge pixel. To illustrate the proposed method, an 8×8 pixels window is showed in Fig. 2. The red pixels are edges detected by the Canny operator. For every edge pixel, the Sobel operators at four directions are used. D1, D2, D3 and D4 are identified as the first directional derivatives of the edge pixels. Then the adaptive thresholds are applied to determine the edge direction, specifically:

- If D1≥D2 and D1≥ T_x , the edge direction at this point is considered at x direction.
- If D2>D1 and D2≥ T_y , the edge direction at this point is considered at y direction.
- If D4≥D3 and D4≥ T_{45} , the edge direction at this point is considered at 45° diagonal direction.
- If D3>D4 and D3≥ T_{135} , the edge direction at this point is considered at 135° diagonal direction.

255	255	255	255	255	255	241	119
255	255	255	255	255	250	127	112
255	255	255	255	255	133	111	117
255	255	255	255	143	110	117	118
255	255	255	160	107	117	118	118
255	255	177	105	117	118	118	118
255	203	102	117	118	118	118	118
228	102	117	118	118	118	118	118

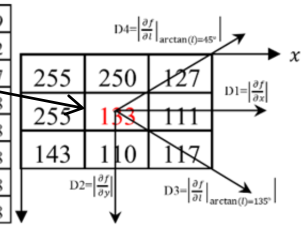


Fig. 3. 8×8 pixels window.

3) Edge-Adaptive Interpolation

The Interpolation is carried out in different regions. For the homogeneous regions, the bilinear interpolation is used. For the heterogeneous regions, the adaptive interpolation is used just the same as [8] (Fig. 3), specifically:

- If the edge is in x direction, then the value $\frac{\text{I}+\text{II}}{2}$ is assigned to a (Fig. 4).
- If the edge is in y direction, then the value $\frac{\text{I}+\text{IV}}{2}$ is assigned to c (Fig. 4).

After horizontal-or-vertical direction interpolation, 45° diagonal-or 135° diagonal direction interpolation is carried out.

- If the edge is in 45° diagonal direction, then the value $\frac{\text{II}+\text{IV}}{2}$ is assigned to b (Fig. 4).
- If the edge is in 135° diagonal direction, then the value $\frac{\text{I}+\text{III}}{2}$ is assigned to b (Fig. 4).

Other points are interpolated by using the bilinear interpolation. This algorithm scales images with scaling factor $F = 2k$, for $k = 1, 2, 3, \ldots$. This could be realized by k iteration. Any size of an image could be obtained by scaling up image with the nearest scaling factor then scaling down the image. This is the main limitation of the algorithm because the proposed scaling scheme is not flexible, it still relies on other down scaling algorithms.

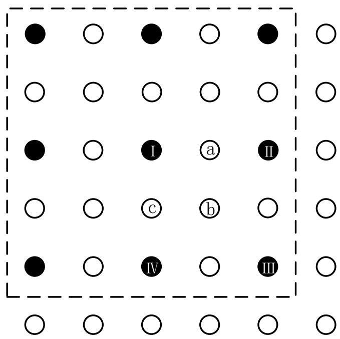


Fig. 4. Scaled image.

C. Canny-Sobel Zig-zag Suppression

The zig-zag usually happens in image scaling up, especially in the edge regions of the scaled image. Thus, a zig-zag suppression method is proposed in our algorithm. The aim of this method is to smooth the image edges.

1) Determining Adaptive Thresholds

This part is the same as the thresholds determination in the Sobel edge-adaptive interpolation, the Canny and Sobel operators are combined to determine the thresholds. For a scaled image, the Canny operators are used at first. Then the Sobel operators are applied at the detected edges and the four thresholds T_x , T_y , T_{45} and T_{135} are founded.

2) Sobel Edge Direction Detection

This part is the same as the edge direction detection that mentioned before, four threshold values T_x , T_y , T_{45} and T_{135} are used. The edge direction will be used at the next part.

3) Zig-zag Suppression

To smooth image edges, the neighbor pixels around the edge are also used. In work [5], the image edges are detected at first, and then the mean value of neighbor pixels is assigned to the pixels which next to the detected edges. This method could suppress zig-zag with low computational complexity. The improved algorithm of it is used in our zig-zag suppression algorithm.

- If the edge is in x direction, just as the red line shown in Fig. 5, then $f(x,y-1) = W\,x \cdot f(x,y-1) + (1 - W\,x) \cdot f(x,y-2)$ and $f(x,y+1) = W\,x \cdot f(x,y+1) + (1 - W\,x) \cdot f(x,y+2)$, where $W\,x$ is the weight and $f(x,y-1)$, $f(x,y-2)$, $f(x,y+1)$, $f(x,y+2)$ are shown in Fig. 5 as the red points.
- If the edge is in y direction, just as the blue line shown in Fig. 5, then $f(x-1,y) = W\,y \cdot f(x-1,y) + (1 - W\,y) \cdot f(x-2,y)$ and $f(x+1,y) = Wy \cdot f(x+1,y) + (1 - Wy) \cdot f(x+2,y)$. The $f(x-1,y)$, $f(x-2,y)$, $f(x+1,y)$, $f(x+2,y)$ are shown in Fig. 5 as the blue points.
- If the edge is in 45° diagonal direction, just as the yellow line showed in Fig. 5, then $f(x-1,y-1) = W45 \cdot f(x-1,y-1) + (1 - W45) \cdot f(x-2,y-2)$ and $f(x+1,y+1) = W45 \cdot f(x+1,y+1) + (1 - W45) \cdot f(x+2,y+2)$. The $f(x-1,y-1)$, $f(x-2,y-2)$, $f(x+1,y+1)$, $f(x+2,y+2)$ are shown in Fig. 5 as the yellow points.
- If the edge is in 135° diagonal direction, just as the green line showed in Fig. 5, then $f(x-1,y+1) = W135 \cdot f(x-1,y+1) + (1 - W135) \cdot f(x-2,y+2)$ and $f(x+1,y-1) = W135 \cdot f(x+1,y-1) + (1 - W135) \cdot f(x+2,y-2)$. The $f(x-1,y+1)$, $f(x-2,y+2)$, $f(x+1,y-1)$, $f(x+2,y-2)$ are shown in Fig. 5 as the green points..

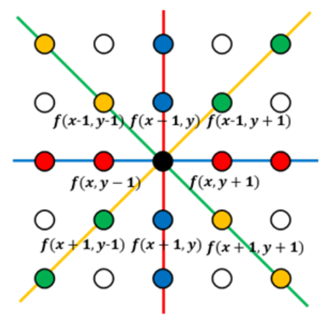


Fig. 5. Zig-zag suppressing algorithm.

The choice of weights W_x , W_y , W_{45} , W_{135} is not simple. $W_x = W_y = W_{45} = W_{135} = 0.5$ is the simplest way. However, this method may blur the boundary between edge and other regions. Therefore, a weighted mean value is proposed in our algorithm, for example, $f(x,y-1) = W_x \cdot f(x,y-1) + (1 - W_x) \cdot f(x,y-2)$, if $f(x,y-1)$ is much bigger, then more weight is added to it. In the proposed algorithm, the weight is set as:

$$W_x = \frac{f(x,y-1)}{f(x,y-1)+f(x,y-2)+1} \text{ and } 1 - W_x = \frac{f(x,y-2)}{f(x,y-1)+f(x,y-2)+1}$$

For the other direction, $f(x,y+1) = W_x \cdot f(x,y+1) + (1 - W_x) \cdot f(x,y+2)$ the weight is set as:

$$W_x = \frac{f(x,y+1)}{f(x,y+1)+f(x,y+2)+1} \text{ and } 1 - W_x = \frac{f(x,y+2)}{f(x,y+1)+f(x,y+2)+1}$$

The same algorithm is applied to every direction of the edges. The output results perform better than the results when $W_x = W_y = W_{45} = W_{135} = 0.5$, especially in the preservation and clarity of edges.

IV. Results and Analysis

The four pictures in Fig. 6 are the images scaled down 2 times from their original images and are used to test the performance of the proposed algorithm. MATLAB is our tool for implementation. The pictures are scaled down two times at first, then scale up four times. The source of images is from HuaWei Honor 8. One can get them from the self-contained album of every HuaWei cell phone. The quality of scaled image is measured by PSNR (Peak Signal-to-Noise Ratio). However, PSNR is not always accurate, therefore, SSIM (Structure Similarity) is also applied for evaluating image quality. The reference images are the original images. Higher PSNR or SSIM value means higher image quality. Images are measured by the two methods independently. Therefore, the efficiency of algorithms could be simply regarded as the performance of PSNR or SSIM. The better algorithm, the higher is the PSNR or SSIM value. Pictures (a) and (b) in Fig. 7 to 10 are the expanded images using the traditional linear and bicubic interpolation, respectively. Pictures (c) and (d) in Fig. 7 to 10 are interpolated by using the proposed algorithm before and after zig-zag suppression, respectively.

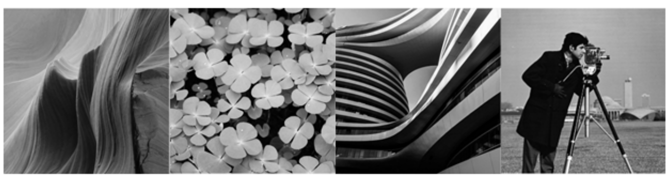
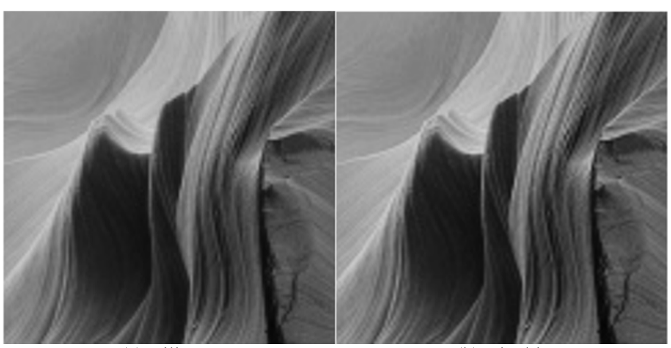
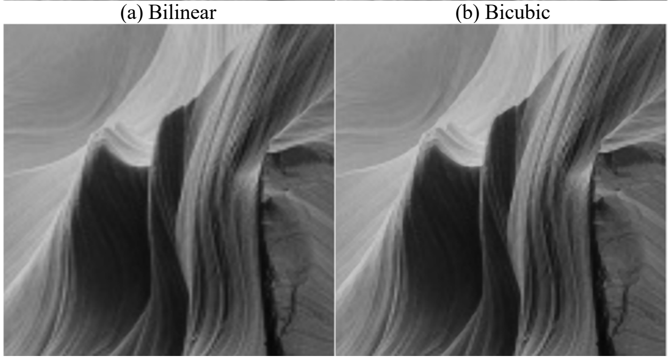


Fig. 6. Pictures from left to right are the scaled test images: a) Patterns, b) Clover, c) Building, d) Cameraman.

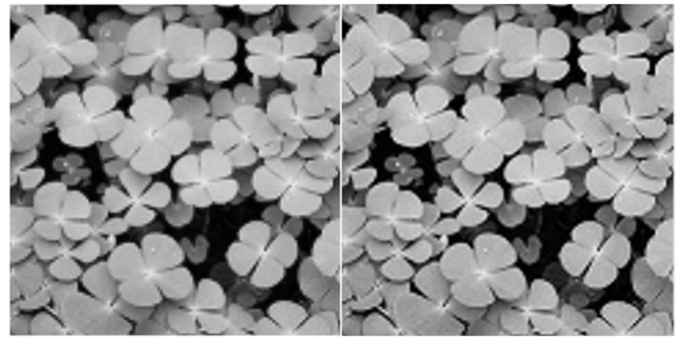


(a) Bilinear (b) Bicubic

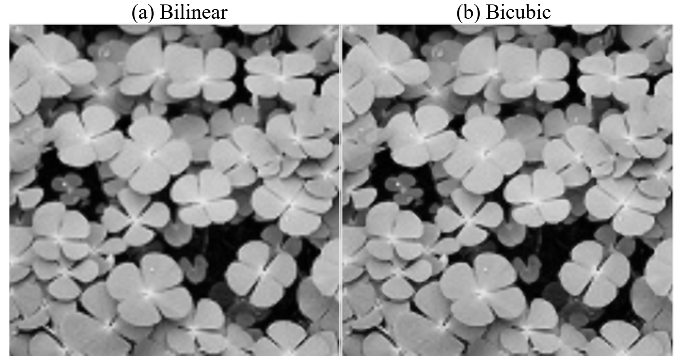


(c) Before Zig-zag Suppression (d) After Zig-zag Suppression

Fig. 7. Image 1: Patterns.

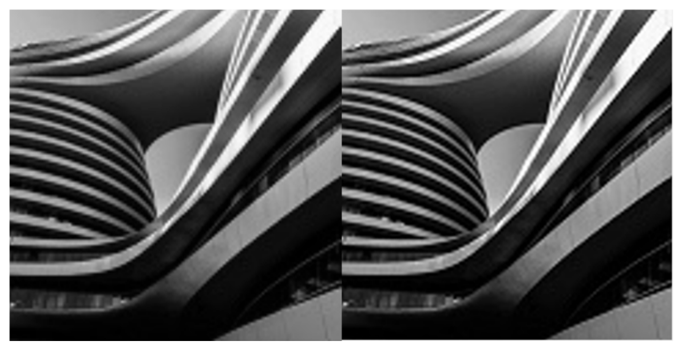


(a) Bilinear (b) Bicubic

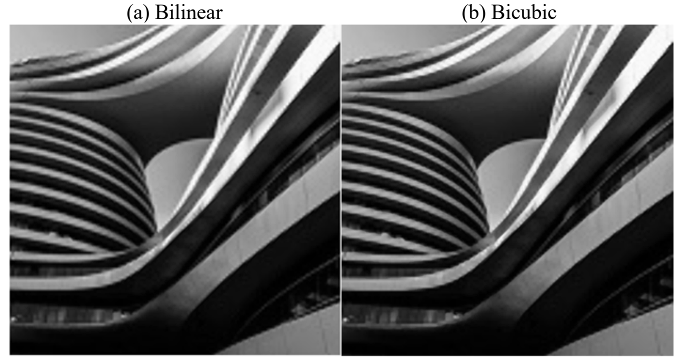


(c) Before Zig-zag Suppression (d) After Zig-zag Suppression

Fig. 8. Image 2: Clover.

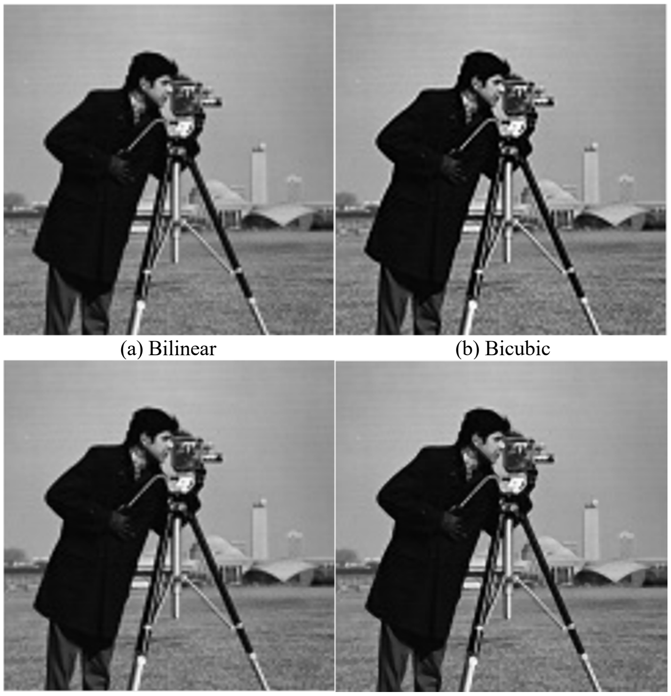


(a) Bilinear (b) Bicubic



(c) Before Zig-zag Suppression (d) After Zig-zag Suppression

Fig. 9. Image 3: Building.



(a) Bilinear (b) Bicubic (c) Before Zig-zag Suppression (d) After Zig-zag Suppression

Fig. 10. Image 4: Cameraman.

The results show that the proposed algorithm preforms better when image has many edges and patterns. The edges of zig-zag suppressed images look more smooth and natural than the image without zig-zag suppression. The PSNR and SSIM index are shown on Table 1. These data illustrated that the proposed algorithm performs better than the bilinear interpolation and sometimes better than bicubic interpolation. The SSIM of zig-zag suppressed image is usually very high, which means the image looks more natural. The PSNR of zig-zag suppressed image usually higher than the image without zig-zag suppression. Our proposed algorithm increases the processing of the diagonal edge directions that was not considered in [5]. Further, we use the weight factor instead of the simply averaging operation. Also, we add zig-zag noise suppression step to increase the image quality.

However, the zig-zag suppression method may not perform well when zig-zag is not obvious. It would have "cartooning" effect which is illustrated in [9].

V. Conclusion

In this paper, we proposed a Canny and Sobel based edge adaptive image scaling algorithm. The results showed that the proposed algorithm performs better than the bilinear interpolation and the mentioned algorithms [5], [8], particular in the presence of many edges and textures. More edge directions and weight factors for zig-zag suppression are added for processing and analysis. So, the proposed algorithm produces images with high resolution and well-preserved edges, consistent with the experimental results.

References

[1] Y. Keller, A. Averbuch and M. Israeli, "Pseudopolar-based estimation of large translations, rotations, and scalings in images", *IEEE Trans. Image Process*, vol. 14, no. 1, pp. 12-22, Jan. 2005.

[2] S. Carrato and L. Tenze, "A high quality 2x image interpolator", *IEEE Signal Processing Letter*, vol. 7, pp. 132-134, Jun. 2000.

[3] R. C. Gonzalez and R. E. Woods, "Digital Image Process," Prentice Hall, NJ, 2002.

[4] R. Keys, "Cubic Convolution Interpolation for Digital Image Processing", *IEEE Trans. Signal Processing*, vol. 29, pp. 1153-1160, 1981.

[5] H. Shi, and R. Ward, "Canny edge based image expansion", Proceedings of IEEE International Symposium on Circuits and Systems - ISCAS 2002, Volume I, 785-788, May 2002.

[6] Y. −T. Lai, C.-F. Tzeng and H.-C. Wu, "Adaptive image Scaling Based on Local Edge Directions", International Conference on Intelligent and Advanced Systems - ICIAS 2010, June 2010, Kuala Lumpur, Malaysia, 2010

[7] W. Jiang, H. Xu, G. Chen W. Zhao and W. Xu, " An Improved Edge-Adaptive Image Scaling Algorithm", 2009 IEEE 8th International Conference on ASIC, pp. 895-897, 2009.

[8] W. Chen, Y. Yu and H. Shi "An Improvement of Edge-Adaptive Image Scaling Algorithm Based on Sobel Operator," IEEE International Conference on Information Science and Control Engineering - ICISCE 2017, July 2017, Changsha, China, 2017

[9] W. Qing, and R. K. Ward. "A New Orientation-Adaptive Interpolation Method." *IEEE Transactions on Image Processing A Publication of the IEEE Signal Processing Society* 16 (4), pp. 889-900, 2007.

TABLE I. Performance of Different Algorithms

PSNR	Different Algorithms					
	Bilinear	***Bicubic***	***W. Chen's Algorithm***	***H. Shi's Algorithm***	***Proposed Algorithm (Before Zig-zag Suppression)***	***Proposed Algorithm (After Zig-zag Suppression)***
Patterns	29.1823	29.1873	29.2612	29.2353	29.2443	**29.3103**
Clover	22.952	22.9661	22.9816	22.9978	22.9958	**23.0829**
Building	22.0184	22.091	22.2097	22.3075	22.1599	**22.532**
Cameraman	26.1713	**26.374**	26.2874	26.2607	26.2192	26.3372
Lena	26.7181	**27.0343**	26.8111	26.8162	26.7988	26.9491
SSIM	**Different Algorithms**					
	Bilinear	***Bicubic***	***W. Chen's Algorithm***	***H. Shi's Algorithm***	***Proposed Algorithm (Before Zig-zag Suppression)***	***Proposed Algorithm (After Zig-zag Suppression)***
Patterns	0.7281	**0.7345**	0.7315	0.7305	0.7304	0.7335
Clover	0.7501	0.7441	0.7525	0.7529	0.7525	**0.7588**
Building	0.7268	0.7233	0.7370	0.7393	0.7331	**0.7502**
Cameraman	0.8155	0.8183	0.8181	0.8171	0.8168	**0.8194**
Lena	0.7781	**0.7885**	0.7815	0.7789	0.7803	0.7828