An Edge-based Adaptive Image Interpolation and Its VLSI Architecture

Hongbin Sun, Fengwei Zhang and Nanning Zheng
Xi'an Jiaotong University, Xi'an, Shaanxi, China
E-mail: hsun@mail.xjtu.edu.cn Tel/Fax: +86-29-82668672

***Abstract*—The design of high quality yet real-time image interpolation has become increasingly important for digital TV SoC, as the size of flat-panel display has been steadily increased . This paper aims to develop a real-time image interpolation algorithm that can achieve the high-ratio image scaling with sharp and natural edges. Comparing with conventional image interpolation approaches that often suffer from either image blurring/jagged problem or high computational cost, this paper proposes a high-efficient edge directional image interpolation approach that support multiple interpolation directions and hence can well preserve the detail and edges. Experimental result shows that the proposed image interpolation algorithm is able to achieve high quality at the high image scaling ratio while only incurring very low computational cost. And the VLSI architecture of proposed image interpolation is also presented.**

I. Introduction

Technology and practice of digital TV are now entering one of the most rapidly changing eras in its history, covering UHD-TV (Ultra High Definition TV), Smart TV, 3D-TV, and many other concepts [1]. One trend in TV technology is to support UHD (4K-TV: 4,096 pixels $\times$ 2,160 pixels; 8K-TV: 7,680 pixels $\times$ 4,320 pixels). During the past decade, the size of flat-panel display has been steadily increased from SDTV to HDTV, and we may see the pervasive use of 4K*2K panel in the very near future. However, the size of video source is not increased at the same pace as the display panel does. This has made it become increasingly important to design high quality image interpolation to display video sources with various size on the same large size panels.

Image interpolation plays an important role in image processing domain, and has been used in digital TV, digital camera and many other applications for image enlargement and local image zooming. The most commonly used image interpolation methods include nearest-neighbor interpolation, linear interpolation and cubic convolution interpolation [2]. However, image artifacts like blurring or zigzag on edge may occur when these interpolation methods are used. Therefore, more advanced interpolation methods have been proposed to improve the quality of interpolated image [3-16]. These interpolation methods are mainly based on explicit or implicit detection of local image features, and employ different interpolation function to different image region. One of the most well know and widely used algorithms is called new edge-directed interpolation (NEDI) [5], which performs good subjective quality by estimating and using local covariance coefficients to adapt the following interpolation. Many other works have been proposed to further improve the quality and computational efficiency of NEDI methods [6], [8], [11], [14]. However, the real-time implementation of NEDI-based methods seems infeasible even for Graphic Processing Unit (GPU) platform.

This paper proposes an efficient edge-based adaptive image interpolation for real-time digital TV application. The basic idea is to leverage advanced edge detection to decide whether the interpolating pixel is an edge pixel or non-edge pixel, then interpolate the edge pixel and non-edge pixel with different interpolation functions. With regarding to edge detection algorithm, we propose an improved Canny edge detection, which is able to guarantee the accuracy and precision with significantly reduced computational complexity. And we continue to use conventional bicubic method to interpolate the non-edge pixels. While for the edge pixels, we adaptively employ bicubic or bilinear function in the square or parallelogram interpolation kernel, according to the detected edge direction. Simulation results demonstrate that, compared with conventional or other edge-based interpolation methods, the proposed adaptive interpolation outperforms at both objective and subjective quality. Moreover, an VLSI architecture design and its hardware cost is also presented, to further demonstrate the computational efficiency of proposed method.

The rest of this paper is organized as follows: Section II briefly discusses the background of image interpolation. Section III describes the proposed edge based adaptive image interpolation algorithm. Section IV gives the VLSI architecture. The experimental results are presented in Section V. Finally, we summarize the contributions of this paper in Section VI.

II. Image Interpolation Overview

As image interpolation is an important technique for video processing systems, various image interpolation methods have been proposed to achieve high-ratio and detail-preserving image scaling. Previous image interpolation methods can be roughly divided into two main categories: (1) Conventional linear interpolation methods that use constant convolution kernels for the entire image, and the value of each new pixel is obtained by computing a linear combination of the values of the original neighboring pixels. (2) Non-linear interpolation methods that are usually based on explicit or implicit detection of local image features, and different interpolation functions are adaptively applied according to the detected local features.

Conventional linear interpolation methods include nearest neighbor interpolation, bilinear interpolation, bicubic interpolation [2] and many other improved algorithms [7], [10]. These methods are computational efficient and hardware friendly, especially the bicubic interpolation (fitting a cubic function on the 16 closest neighbors) provides visually good images. However, these interpolation methods tends to smooth the detail and only keep low frequency content in the processed image. As they are not able to enhance the high frequencies or preserve the edges equally well, they may produce some annoying visual problems, such as aliasing, blurring or other artifact.

Non-linear methods tends to solve the problems introduced by linear interpolation. One of the most well known non-linear algorithms is called NEDI, which use the covariant geometric regularity to achieve high quality image scaling [5]. Many works have been proposed to further improve the quality and efficiency of NEDI method. However, these NEDI-based methods [5], [6], [8], [11], [14] are restricted to an scaling ratio of 2^n, where n is an integer. More important, this kind of methods inevitably incur high computational cost, and are difficult to realize real-time implementation for digital TV application.

Therefore, realistic high quality is not the only issue to be considered in choosing an image interpolation, the computational efficiency of the methods should also be taken into account, especially in the case of real-time applications. Ref. [3], [15], [16] present several relatively less complex non-linear interpolation, which detect the edge pixels and their edge directions first, then interpolate non-edge pixels with conventional linear interpolation method, while interpolate the edge pixels along the detected edge direction with the neighboring pixels. Their basic processing flow can be summarized and illustrated in Fig. 1. This kind of edge directed non-linear image interpolation methods is able to provide decent quality with the reasonable computational cost for hardware design, and hence is what we mainly concern in this paper.

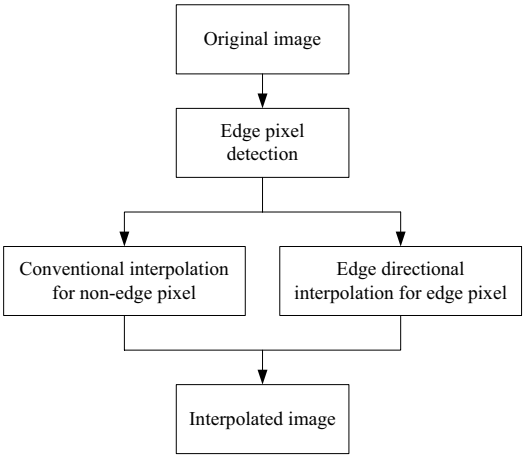


Fig. 1. Basic processing flow of edge directional image interpolation.

III. Proposed Adaptive Image Interpolation Algorithm

This section presents our proposed edge directed adaptive image interpolation algorithm, where the key idea is to leverage more advanced edge detection method to classify edge pixels from other pixels, and further adaptively apply different interpolation functions to both edge and non-edge pixels according to their edge directions.

A. Improved Canny Edge Detection

Edge detection is the most critical component for edge directed image interpolation. Previous edge detection methods often use relatively simple approaches that often provide low accuracy and very few edge directions, e.g. sobel operator, as more advanced edge detection tends to dramatically increase computational requirement and hence complicate the hardware design. However, this tends to seriously limit the quality of following interpolation procedure, as edge directional interpolation depends on the accuracy and precision of edge detection. Canny edge detection seems to be an competitive candidate [4]. However, the conventional Canny detection uses double thresholding algorithm to detect and link edges, hence inevitably incurs computationally intensive iterative operations. In this paper, we propose an improved Canny edge detection method that can guarantee the accuracy and precision of detected edge while still maintain reasonable computational and hardware complexity.

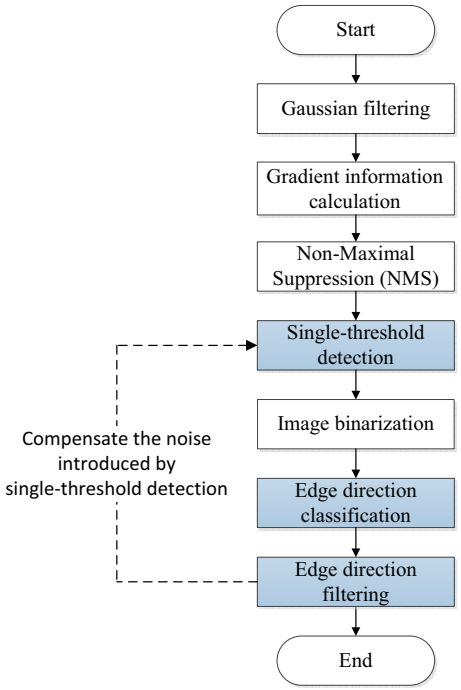


Fig. 2. Flow chart of improved Canny edge detection algorithm.

The flow chart of proposed edge detection algorithm is illustrated in Fig. 2. In general, the proposed algorithm follows the conventional Canny edge detection, which first smoothes

the image with a Gaussian filter, then computes the gradient magnitude and orientation using approximations of partial derivatives, thins edges by applying Non-Maximal Suppression (NMS), and finally detects edges with threshold comparison. Compared with conventional Canny edge detection, the proposed edge detection algorithm improves at the following three major part:

- The single thresholding is applied instead of double thresholding algorithm, to avoid the computational complexity and hardware design challenge incurred by iterative calculation of double thresholding in conventional Canny algorithm.
- In contrast to conventional Canny algorithm that assigns the edge direction at any angle, the proposed edge detection separates the edge direction into eight different direction regions by using edge direction classification.
- To compensate the detection noise introduced by single thresholding, we further propose to leverage edge direction filtering to detect the edge pixels more precise and only reserve the regular and confidential edge.

As shown in Fig. 3, the detected edge directions are separated into eight different direction regions. And these direction regions are further classified into three different edge direction classes. The reason to further classify edge direction regions is that different edge direction class will require utterly different interpolation function as explained in the following subsection. We note that all the selected angle of edge direction in proposed edge direction region can be located in the pixel array, therefore the following edge directed interpolation can be completed with existing pixels in the original image.

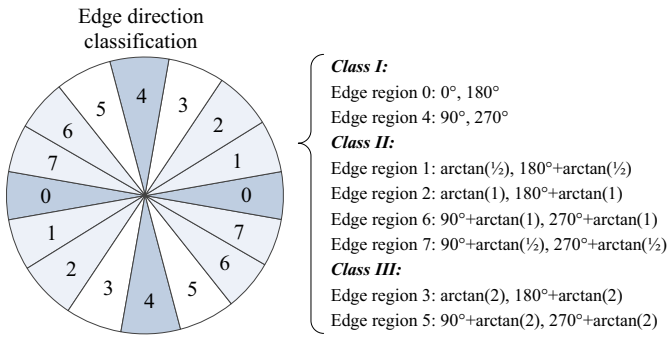


Fig. 3. Edge region and edge direction classification.

Although single thresholding is able to dramatically reduce the computational complexity, it tends to introduce detection noise that may have a negative impact on the following interpolation operation and hence incur unacceptable artifact. To avoid this kind of artifact, we propose to use edge direction filtering to compensate the single thresholding, as illustrated in Fig. 2. The edge direction filtering can be explained as follows: Let pi represents the interpolating pixel and P(m,n) represents one of the neighboring 16 original pixels that are close to pixel pi, as shown in Fig. 4. If the number of edge pixels in the pixel array is larger than threshold_edge and the majority of edge pixels belonging to the same direction region, we set the interpolating pixel pi as edge pixel; otherwise, the interpolating pixel pi is set as non-edge pixel, even though it has been detected as edge pixel after single threshold edge detection.

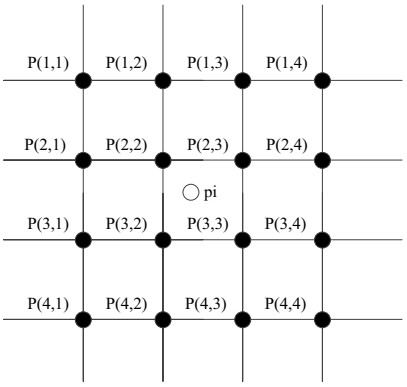


Fig. 4. Pixel array for edge direction filtering.

B. Edge-Directed Interpolation

Due to the improved Canny edge detection described in the above subsection, pixels in the overall image can be classified into two major categories: edge and non-edge pixels. The interpolation function of non-edge pixels seems simple, as it just follows the conventional cubic convolution methods. However, the interpolation functions of edge pixels vary according to the edge direction of each individual pixels.

For the pixels of which edge directions belong to *Class I*, the interpolation function is the same as conventional bicubic interpolation, as the edge direction is either horizontal or vertical. As shown in Fig. 5, Let pi represent the interpolated pixel, $P(i+m, j+n)$ represent the original pixel participated in the interpolation, m and n stand for the horizontal and vertical indexes to the referenced original pixel $P(i, j)$, respectively. The bicubic interpolation function can be represented as follows.

$$pi(x, y) = \sum_{m=-1}^{2} \sum_{n=-1}^{2} P(i+m, j+n) * C_{m+1}(\Delta h) * C_{n+1}(\Delta v). \quad (1)$$

Where $C_0(u) = (u^3+2\times u^2-u)/2$, $C_1(u) = (3\times u^3+5\times u^2+2)/2$, $C_2(u) = (-3\times u^3+4\times u^2+u)/2$, $C_3(u) = (u^3-u^2)/2$, $\Delta h = x - x_i$ and $\Delta v = y - y_j$.

We can see that the interpolated pixel is calculated by using the 16 neighboring pixels, and Δh and Δv represent the horizontal and vertical distances between the interpolated and reference pixels. The shape of interpolation kernel is square, consisted of 16 neighboring pixels. Hence, in this paper, we refer this kind of interpolation to as square-bicubic interpolation. And the square-bicubic interpolation is just the same as what we use in conventional bicubic interpolation.

For the interpolating pixels belonging to *Class II&III*, we can not continue to use the square-bicubic interpolation, as

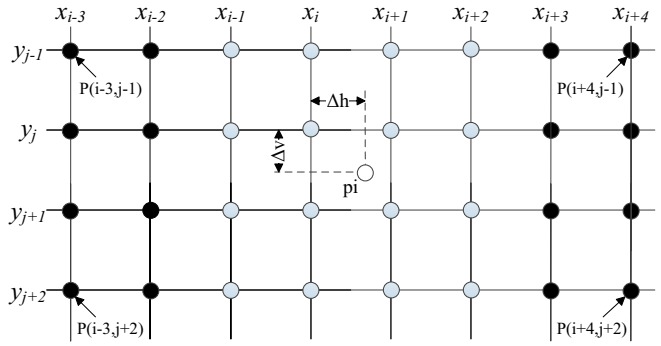


Fig. 5. Example of square-bicubic interpolation for pixels belonging to *Class I*.

do not interpolate along the edge direction may cause jagged or zigzagging artifacts. Instead, we leverage a parallelogram interpolation kernel to well preserve the edge regions. Take *Class II* for example, the parallelogram interpolation kernel is quite similar to its square interpolation counterpart, except that the interpolated pixel is generated by using the 16 neighboring pixels in the adjacent parallelogram, as shown in Fig. 6. In this paper, we refer it to as parallelogram-bicubic interpolation. And its interpolation function can be calculated as the following equation.

$$pi(x,y) = \sum_{P(i+m,j+n)\in\Phi} P(i+m,j+n) * C_{m+n+1}(\Delta h) * C_{n+1}(\Delta v'). \tag{2}$$

Where Φ is defined as the pixel set that contains all the pixels belonging to the parallelogram kernel. $C_0(u)$, $C_1(u)$, $C_2(u)$ and $C_3(u)$ are defined the same as Equ. 1. And Δh and $\Delta v'$ represent the normalized horizontal and vertical distances between the interpolated and reference pixels.

$$\begin{aligned} \Delta h &= (x - x_i) + (y - y_j)/\tan\theta \\ \Delta v' &= \Delta v * \sin\theta = ((y - y_j)/\sin\theta) * \sin\theta = y - y_j \end{aligned} \tag{3}$$

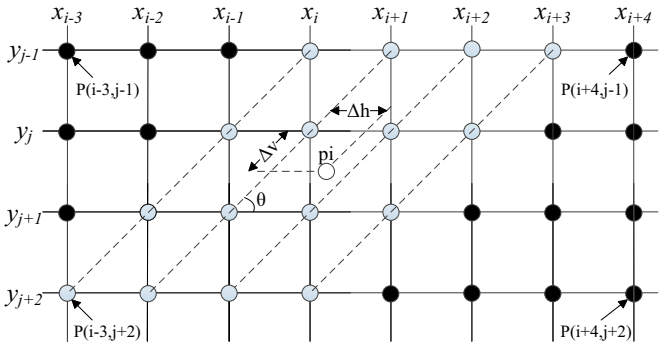


Fig. 6. Example of parallelogram-bicubic interpolation for pixels belonging to *Class II*.

For the interpolating pixel belonging to *Class III*, if we continue to use parallelogram-bicubic interpolation, the hardware cost of interpolation design tends to be significantly increased. As the direction angle is large and bicubic interpolation requires at least 4 pixels along the direction, the required memory storage will be doubled. Therefore, we use parallelogram-bilinear interpolation for pixels belonging to *Class III*, as shown in Fig. 7. The interpolating pixel is generated by using the neighboring 4 pixels in the adjacent parallelogram region. And the interpolation function can be calculated by the following equation.

$$\begin{aligned} &pi(x,y) = \\ &(1-\Delta v)(1-\Delta h')P(i,j) + (1-\Delta v)\Delta h' P(i+1,j-2) + \\ &\Delta v(1-\Delta h')P(i,j+1) + \Delta v \Delta h' P(i+1,j-1). \end{aligned} \tag{4}$$

Where $\Delta h'$ and Δv represent the normalized horizontal and vertical distances between the interpolated and the reference pixels, and are defined as follows.

$$\begin{aligned} \Delta h' &= \Delta h * \cos\theta = ((x - x_i)/\cos\theta) * \cos\theta = x - x_i \\ \Delta v &= (y - y_j) + (x - x_i) * \tan\theta \end{aligned} \tag{5}$$

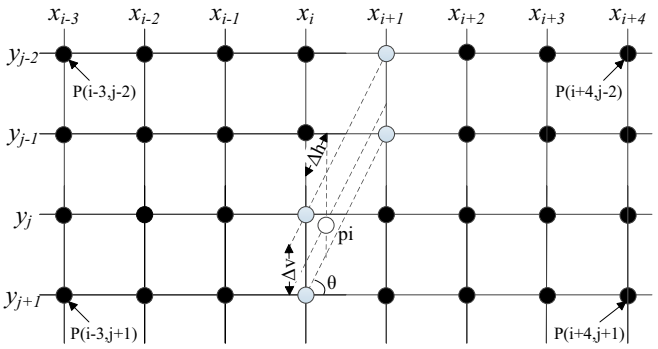


Fig. 7. Example of parallelogram-bilinear interpolation for pixels belonging to *Class III*.

The adaptive use of square-bicubic, parallelogram-bicubic and parallelogram-bilinear kernels in interpolation algorithm, according to the detected edge direction, can significantly improve the interpolation quality of edge regions. Moreover, the computational complexity and hardware design cost has not be dramatically increased in the meanwhile. This is very important for real-time implementation and the following VLSI architecture design.

IV. VLSI Architecture

Compared with conventional non-linear interpolation methods that often involve iterative calculation and require high computational capability, the proposed method is friendly for hardware design. Fig. 8 shows the the VLSI architecture design of proposed edge-based adaptive interpolation algorithm. It includes the color space conversion (CSC), the input and output sync controller, the improved Canny edge detector, the image interpolation and on-chip line buffer memories. Due to the algorithm optimization, the proposed VLSI architecture only employs four on-chip line buffer memories to realize the adaptive multi-kernel bicubic/bilinear interpolation.

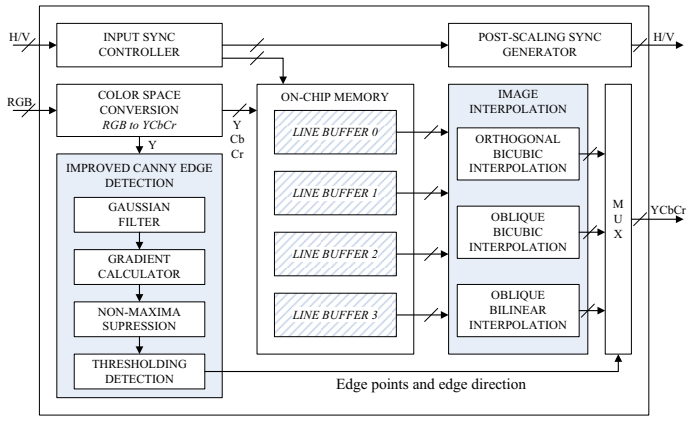


Fig. 8. VLSI architecture of proposed edge-based adaptive image interpolation.

We further implement the VLSI architecture of proposed method to a Xilinx FPGA of Virtex5 XC5VLX330-2. According to the synthesis result, the proposed adaptive interpolation design takes about 36% slice LUT (76,048 LUTs) and 2,772 KB BlockRAM. The critical path delay is 3.1 ns, which means the maximum data rate can reach up to 300MHz. As the cell-based ASIC implementation is significantly faster than FPGA, the proposed design will certainly be able to seamlessly support the real-time video display processing application including HDTV and 4K-TV.

V. Experimental Result

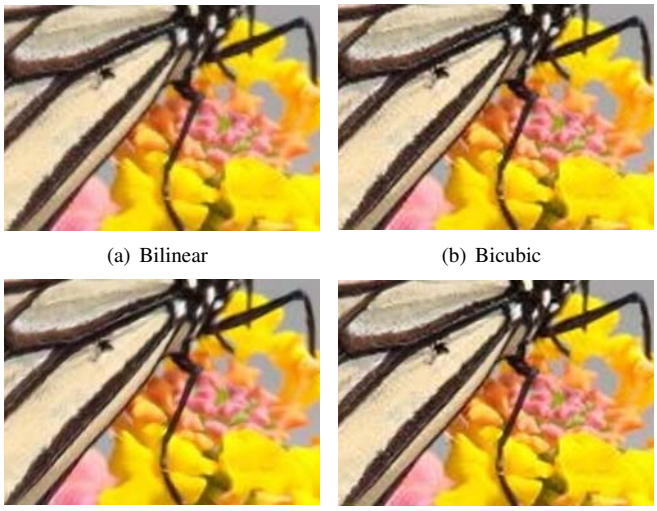
To evaluate the proposed algorithm, we use an image database of 10 natural images selected from morgueFile online archive (http://morguefile.com). All images are subject to the license agreement available at the Web page http://morguefile.com/archive/terms.php. Selected files were RGB color images with a depth of 8 bits per channel. During the upscaling proces, these images are converted to YUV color space, and the interpolation coefficients are computed on the image brightness and are used for all the YUV channels. We performed both subjective and objective tests in order to compare quantitatively the quality of the images created with different methods.

The objective test compares images obtained by downsampling the original images and then enlarging them with different methods. And we use peak signal noise ratio (PSNR) as the objective comparison metric. The proposed edge-based adaptive interpolation is compared with both well known linear methods, i.e. Bilinear and Bicubic [2], and nonlinear methods, i.e. NEDI [5]. Table I shows the PSNR comparison results among the proposed and reference methods. We can see that the PSNR of proposed interpolation method outperforms the other three methods, and is even better than NEDI.

It should be noted that methods with high PSNR are not necessarily corresponding to better visually perceived quality. The major purpose of proposed algorithm is to achieve real-time interpolation and to have better subjective quality. Therefore, we further compare the subjective quality between

TABLE I
PSNR Comparison among Different Methods.

Images	Bilinear	Bicubic	NEDI	Proposed
Zebra	27.71	29.00	29.15	29.16
Butterfly1	29.90	30.89	30.13	30.48
Parrot	29.57	30.14	29.87	30.18
Newspaper	25.78	26.27	26.08	26.09
Greenery	24.54	25.00	24.99	25.14
Crayon	29.38	30.32	30.09	30.81
Eagle	30.80	31.27	31.54	31.50
Butterfly2	33.07	34.08	33.03	34.56
Sunflower	36.53	37.73	37.15	38.41
Ballon	40.28	41.58	41.26	42.23



(a) Bilinear (b) Bicubic

(c) NEDI (d) Proposed

Fig. 9. Subjective comparison of “Flower” image.

proposed and reference methods over the selected 10 natural images. Fig. 9 and 10 show two examples of simulation results. We can conclude that the perceived quality of proposed adaptive interpolation is much better than Bilinear and Bicubic methods, especially for the edge and detail regions. And, it has also achieve the comparable subjective quality to NEDI methods, while the computational complexity is significantly reduced. More important, the proposed interpolation algorithm can further effectively avoid the ‘cartoon’ artifact incurred by NEDI, as shown in Fig. 10.

VI. Conclusions

This paper proposes an edge-based adaptive image interpolation algorithm for real-time digital TV application. The proposed method exploits an improved Canny method to enable the robust and precise edge detection with relatively much lowered computational cost. Furthermore, we adaptively employ the square-bicubic, parallelogram-bicubic and parallelogram-bilinear interpolation according to the detected edge direction, to prevent from the jagged or zigzagging edge. Compared with conventional real-time edge directed interpolation methods, the proposed method can achieve the sharper and more natural

(a) Bilinear (b) Bicubic

(c) NEDI (d) Proposed

Fig. 10. Subjective comparison of "Newspaper" image.

edges after image upscaling with reasonable computational complexity.

VII. Acknowledgments

This research was funded in part grants from the National Science and Technology Major Project of China (No. 2009ZX01033-001-010).

References

[1] F.-L. Luo, W. Williams, R. M. Rao, R. Narasimha and M. J. Montpetit, "Trends in signal processing applications and industry technology," *IEEE Signal Processing Magazine,* vol. 29, no. 1, pp. 184, Jan. 2012.

[2] R. Keys, "Cubic convolution interpolation for digital image processing," *IEEE Trans. Signal Processing,* vol. 29, pp. 1153-1160, 1981.

[3] Y.-T. Lai, C.-F. Tzeng and H.-C. Wu, "Adaptive image scaling based on local edge directions," *IEEE International Conference on Intelligent and Advanced Systems (ICIAS),* pp. 1-4, Jun. 2010.

[4] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. PAMI-8, no. 6, pp. 679-698, Nov. 1986.

[5] X. Li, T. O. Michael, "New edge-directed interpolation," *IEEE Transactions on Image processing,* vol. 10, no. 10, pp. 1521-1527, Oct. 2010.

[6] M. Zhao, G, de Haan, "Content adaptive video up-scaling," *Proceedings of ASCI,* pp. 151-156, 2003.

[7] W. H. Jung, S. L. Hwang, "Adaptive image interpolation based on local gradient features" *IEEE Signal Processing,* vol. 11, no. 3, pp. 359-362, Mar. 2004.

[8] M. Darian, W. P. Thomas, "Adaptively quadratic (AQua) image interpolation," *IEEE Transactions on image processing,* vol. 13, no. 5, pp. 690-698, May 2004.

[9] D. D. Muresan, "Fast edge directed polynomial interpolation," *IEEE International Conference on Image Processing (ICIP),* vol. 2, pp. II-990-3, Sep. 2005.

[10] J.-Z. Shi, S. E. Reichenbach, "Image interpolation by two-dimensional parametric cubic convolution," *IEEE Transactions on image processing,* vol. 15, no. 7, pp. 1857-1870, Jul. 2006.

[11] N. Asuni, A. Giachetti, "Accuracy improvements and artifacts removal in edge based image interpolation," *Proc. of 3rd Intl. Conf. on Computer Vision Theory and Applications,* vol. 1, pp. 58-65, 2008.

[12] H. Kim, S. Park, J. Wang, Y. Kim and J. Jeong, "Advanced bilinear image interpolation based on edge features," *First International Conference on Advances in Multimedia (MMEDIA),* pp. 33-36, Jul. 2009.

[13] J. Chang, D. Yoo and J. Park, "Edge directional interpolation for image upscaling with temporarily interpolated pixels," *Electronics Letters,* vol. 47, no. 21, pp. 1176-1178, 2011.

[14] A. Giachetti and N. Asuni, "Real-time artifact-free image upscaling," *IEEE Transactions on Image Processing,* vol. 20, no. 10, pp. 2760-2768, Oct. 2011.

[15] M.-J. Chen, C.-H. Huang and W.L. Lee, "A fast edge-oriented algorithm for image interpolation," *Elsevier Journal of Image Vision Computing,* vol. 23, no. 9, pp. 791-798, 2005.

[16] Q. Wang and R. Ward, "A new orientation-adaptive interpolation method," *IEEE Transaction on Image Processing,* vol. 16, no. 4, pp. 889-900, Apr. 2007.