International Journal of Innovation and Applied Studies
ISSN 2028-9324 Vol. 3 No. 3 July 2013, pp. 813-819
© 2013 Innovative Space of Scientific Research Journals
http://www.issr-journals.org/ijias/

Color Image Interpolation using Optimal Edge Detector

R. Jansi, Asnath Victy Phamila, and R. Amutha

Department of Electronics and Communication Engineering,
SSN College of Engineering,
Chennai, Tamil Nadu, India

ABSTRACT: This paper proposes an efficient image interpolation algorithm using optimal edge detector. The proposed interpolation algorithm is done in two steps. In the first step, the missing pixels with diagonal neighbors are interpolated and in the second step, the missing pixels with axial neighbors are interpolated. In both the steps classification of edge and smooth pixels is performed using canny edge detector. For the pixels classified as edge pixels, the direction of the edge is found using various structuring elements. The edge pixels are then interpolated along the directional orientation of the edges. The smooth pixels are interpolated using proportionate variation based interpolation technique in which more weights are assigned for the direction with minimum variation. The proposed interpolation algorithm is applied in the NTSC color space for color images. Conventional image interpolation algorithms like nearest neighbor and bi-cubic interpolation algorithms produces artifacts like edge blurring, zig- zag effects etc. The proposed interpolation algorithm produces super resolution images with improved image quality and less distortion. Experimental results show that in addition to the significant increase in visual effects, this algorithm also manifests improvements in quantitative analysis. Quantitative analysis is done using metrics like PSNR and correlation coefficient. From this analysis it is evident that the proposed algorithm performs better than conventional interpolation algorithms.

KEYWORDS: Canny, gradient, interpolation, resolution, structuring element, zoom.

1 INTRODUCTION

Interpolation in image processing is a method to increase the number of pixels in a digital image. Interpolation has been widely used in many image processing applications such as facial reconstruction, multiple description coding, and super resolution. Interpolation based super resolution has been used for a long time, and many interpolation techniques have been developed to increase the quality of this task [1]-[6]. There are three well known interpolation techniques namely, nearest neighbor interpolation, bilinear interpolation, and bi-cubic interpolation. Bi-cubic interpolation is more sophisticated than the other two techniques but produces smoother edges than nearest neighbor and bilinear interpolation.

Xin Li et al, proposed a hybrid approach for interpolation using switching between bilinear interpolation local covariance based adaptive interpolation. In this method local covariance was first calculated for the low resolution image. These covariance values were then used for interpolating the low resolution images using the property of geometric duality between the low resolution covariance and the high resolution covariance [7].

Zhou Dengwen et al, proposed a technique for color image interpolation. In this technique the edge pixels were distinguished from the smooth pixels based on a preset threshold value and then the pixels were interpolated using extended bi-cubic convolution algorithm [8].

Jinglun Shi proposed an algorithm for image interpolation using variation based approach. In this algorithm the pixels of low resolution image were projected using various angular orientations and the sum of absolute difference at various orientations were calculated. The pixels were then interpolated along the direction of minimum variation [9].

In this paper a new approach for image interpolation is proposed. Here the edge pixels are first identified using canny operator. Then using various shapes of structuring elements the direction of edges are identified. Then the edge pixels are interpolated along the direction of minimal variation. The smooth pixels are interpolated using all the neighborhood pixels, with more weightage to the pixels in the direction of minimal difference.

The rest of this paper is organized as follows. Section 2 presents the proposed interpolation algorithm. Section 3 explains the interpolation algorithm for every layer of NTSC color space. Section 4 presents the experimental results and the conclusion is presented in Section 5.

2 PROPOSED COLOR IMAGE INTERPOLATION ALGORITHM

In the proposed method color image interpolation is done using the transformation from RGB to NTSC color space. Interpolation is performed in the NTSC color space and the inverse color transformation returns the interpolated image in the RGB color space. The various test images used for color image interpolation are shown in Fig. 1.

The algorithm for interpolation of every layer of NTSC color space is as follows:

1. Insertion of zeros after every row and column of the image to be interpolated.
2. Apply canny operator to the image after zeros insertion.
3. Classify the pixels as smooth and edge pixels.
4. Interpolate the missing pixels (pixels with the available diagonal neighbors) based on their angular orientation.
5. For the second iteration, repeat steps 1 to 4, to interpolate pixels with axial neighbors.

The same steps are repeated for all the three layers, and inverse transformation of NTSC to RGB conversion yiels the interpolated high resolution image as output.

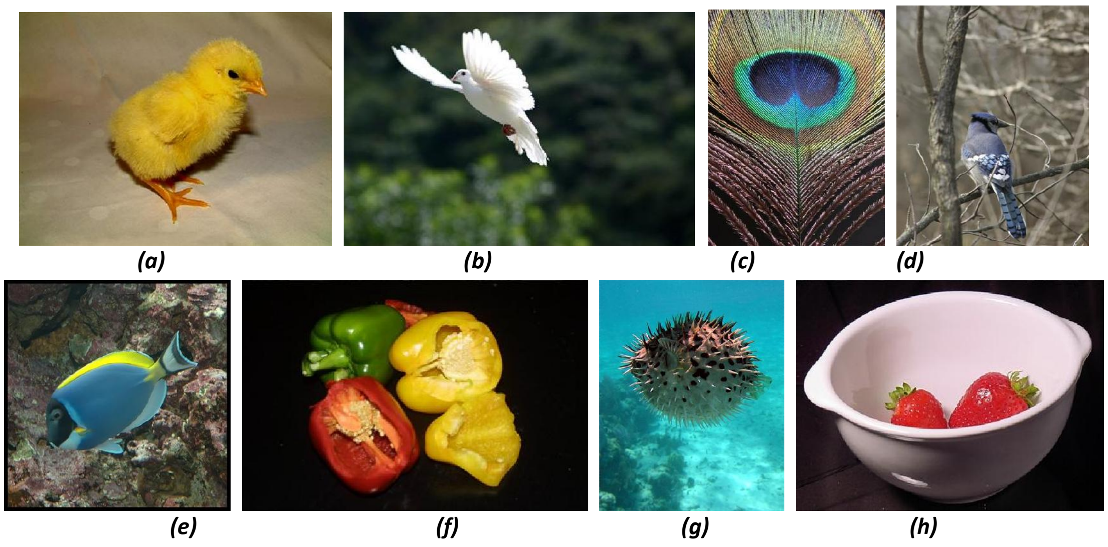

Fig. 1. Set of test images. (a) Chick; (b) Dove; (c) Feather; (d) Bird; (e) Fish; (f) Capsicum; (g) Aqua; (h) Strawberry

3 INTERPOLATION ALGORITHM FOR EVERY LAYER OF NTSC COLOR SPACE

The various steps explained below are performed for the interpolation of every layer of NTSC color space. The three interpolated layers are then converted back to RGB color space to get the interpolated color image. These transformations are performed using Matlab's 'RGB2NTSC' and 'NTSC2RGB' function.

3.1 ZEROS INSERTION

Zeros are inserted after every rows and column of the low resolution image. These zero values are interpolated using the proposed algorithm to yield a good quality high resolution image.

3.2 ITERATION – 1

3.2.1 PIXEL CLASSIFICATION

This section explains steps 2 and 3 of the above mentioned algorithm.

The pixels of gray scale low resolution images are first classified into edge and smooth pixels. Edge pixels are those pixels with abrupt change in the gray scale value. Smooth pixels are those pixels with gradual changes or change in the gray scale value. Two different methods are adopted for interpolating the edge and the smooth pixels.

The proposed algorithm for the classification of edge and smooth pixels is as follows:

- Insert zeros after every row and column of the low resolution image using (1).

$$Q(2i,2j)=P(i,j) \tag{1}$$

  where *Q* represents the image to be interpolated and *P* represents the low resolution image.
- Apply canny operator (optimal edge detector) and extract the edge of the image *Q*. This gives the gradient image *G*1 as output.
- Apply the following 3×3 structuring element with the centre as the pixel to be interpolated to the gradient image *G*1.

$$\begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 1 \end{bmatrix} \tag{2}$$

- There are 3 cases possible

Case 1. If the output of the structuring element is one of the following, the pixel is classified as an edge pixel oriented along 45^0.

$$\begin{bmatrix} 0 & * & 1 \\ * & * & * \\ 1 & * & 0 \end{bmatrix} \text{ (or) } \begin{bmatrix} 1 & * & 1 \\ * & * & * \\ 1 & * & 0 \end{bmatrix} \text{ (or) } \begin{bmatrix} 0 & * & 1 \\ * & * & * \\ 1 & * & 1 \end{bmatrix} \tag{3}$$

Case 2. Else if the output of the structuring element is one of the following, the pixel is classified as an edge pixel oriented along 135^0.

$$\begin{bmatrix} 1 & * & 0 \\ * & * & * \\ 0 & * & 1 \end{bmatrix} \text{ (or) } \begin{bmatrix} 1 & * & 1 \\ * & * & * \\ 0 & * & 1 \end{bmatrix} \text{ (or) } \begin{bmatrix} 1 & * & 0 \\ * & * & * \\ 1 & * & 1 \end{bmatrix} \tag{4}$$

Case 3. The pixels that do not satisfy the above two conditions are classified as smooth or non- edge pixels.

3.2.2 INTERPOLATION OF PIXELS WITH DIAGONAL NEIGHBORS

The edge pixels are interpolated along the direction of the edges and the smooth pixels are interpolated using all the neighborhood pixels. This section explains step 4 of the above mentioned algorithm.

An overlapping window of size 7×7 is moved over the image *Q* with the pixel '*X*' to be interpolated as the center. Fig. 2. represents the 7×7 window with the center pixel 'X' to be interpolated and its neighboring pixels P_1-P_{16}.

The edge pixels oriented along 45^0 are interpolated using (5).

$$X_{45^0} = \frac{-2*P_{13}+14*P_{10}+14*p_7-2*P_4}{24} \tag{5}$$

Similarly, the edge pixels oriented along 135^0 are interpolated using (6).

$$X_{135^0} = \frac{-2*P_1+14*P_6+14*p_{11}-2*P_{16}}{24} \tag{6}$$

P_1	0	P_2	0	P_3	0	P_4
0	0	0	0	0	0	0
P_5	0	P_6	0	P_7	0	P_8
0	0	0	**X**	0	0	0
P_9	0	P_{10}	0	P_{11}	0	P_{12}
0	0	0	0	0	0	0
P_{13}	0	P_{14}	0	P_{15}	0	P_{16}

Fig. 2. ***A 7X7 window showing the center pixel, 'X' to be interpolated and its neighbors***

The pixels which were classified as smooth pixels are interpolated using (7)

$$X = (\alpha * X_{45^0}) + (1-\alpha) * X_{145^0} \tag{7}$$

where,

$$\alpha = \frac{S_{135^0}}{S_{45^0} + S_{135^0}} \tag{8}$$

$S_{45}{}^{0}$ represents the sum of absolute difference of the 4 pixels along 45^0 and $S_{135}{}^{0}$ represents the sum of absolute difference of the 4 pixels along 135^0.

3.3 ITERATION - 2

In the second iteration, the pixels with the axial neighbors are estimated. This section explains step 5 of the proposed algorithm.

3.3.1 PIXEL CLASSIFICATION

The following algorithm is proposed for the classification of edge and smooth pixels in the second pass:

- The partially interpolated high resolution image, *R* obtained after the first iteration is again passed through the canny operator for edge detection. This gives the gradient image *G*2 as output.
- Apply the following 3×3 structuring element with the centre as the pixel to be interpolated to the gradient image *G*1.

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{9}$$

- There are 3 cases possible

Case 1. If the output of the structuring element is one of the following, the pixel is classified as an edge pixel oriented along horizontal direction.

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & * & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad \text{(or)} \quad \begin{bmatrix} 0 & 0 & 0 \\ 0 & * & 0 \\ 1 & 1 & 1 \end{bmatrix} \tag{10}$$

Case 2. Else if the output of the structuring element is one of the following, the pixel is classified as an edge pixel oriented along vertical direction.

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & * & 0 \\ 1 & 0 & 0 \end{bmatrix} \quad \text{(or)} \quad \begin{bmatrix} 0 & 0 & 1 \\ 0 & * & 1 \\ 0 & 0 & 1 \end{bmatrix} \qquad (11)$$

Case 3. The pixels that do not satisfy the above two conditions are classified as smooth or non- edge pixels.

3.3.2 Interpolation Of Pixels with Axial Neighbors

Similar to iteration 1, the edge pixels are interpolated along the direction of the edges, and the smooth pixels are interpolated using all the available neighborhood pixels.

An overlapping window of size 7×7 is moved over the partially interpolated image *R*, with the pixel '*Y*' to be interpolated as the center. Fig. 3. represents the 7×7 window with the center pixel 'Y' to be interpolated and its neighboring pixels P_1-P_{25}.

The edge pixels oriented along horizontal direction or 180^0 are interpolated using (12).

$$X_{180^0} = \frac{-2 * P_{12} + 14 * P_{13} + 14 * p_{14} - 2 * P_{15}}{24} \qquad (12)$$

Similarly, the edge pixels oriented along vertical direction or 90^0 are interpolated using (13).

$$X_{90^0} = \frac{-2 * P_2 + 14 * P_{10} + 14 * p_{17} - 2 * P_{24}}{24} \qquad (13)$$

0	P_1	0	P_2	0	P_3	0
P_4	0	P_5	0	P_6	0	P_7
0	P_8	0	P_{10}	0	P_{11}	0
P_{12}	0	P_{13}	**Y**	P_{14}	0	P_{15}
0	P_{16}	0	P_{17}	0	P_{18}	0
P_{19}	0	P_{20}	0	P_{21}	0	P_{22}
0	P_{23}	0	P_{24}	0	P_{25}	0

Fig. 3. A 7X7 window showing the center pixel, 'Y' to be interpolated and its neighbors

The pixels which were classified as smooth pixels are interpolated using (14)

$$Y = (\beta * X_{180^0}) + (1-\beta) * X_{90^0} \qquad (14)$$

where,

$$\beta = \frac{S_{90^0}}{S_{90^0} + S_{180^0}} \qquad (15)$$

$S_{90}{}^0$ represents the sum of absolute difference of the 4 pixels along 90^0 and $S_{180}{}^0$ represents the sum of absolute difference of the 4 pixels along 180^0.

4 Experimental Results

The efficiency of the proposed interpolation algorithm was quantitatively measured using correlation coefficient and color peak signal to noise (CPSNR) ratio. The color images were decimated by a factor of 2 and the proposed interpolation algorithm was applied and was the images were interpolated. The images were also interpolated using conventional interpolation algorithms like nearest neighbor and bi-cubic interpolation algorithm. The quantitative measurement metrics was calculated using the ground truth image and the interpolated images and the algorithms were compared. This comparison is shown in Table 1 and Table 2. From these tables it is evident that the proposed algorithm outperforms the conventional interpolation algorithms. Fig. 4. Shows Dove image interpolated by a factor of 2, using various interpolation algorithms.

Table 1. Comparison of Correlation coefficient

Test Images	**Correlation Coefficient**		
	Nearest Neighbor	**Bi-Cubic**	**Proposed**
Chick	0.786	0.882	0.924
Dove	0.794	0.957	0.975
Feather	0.912	0.922	0.963
Fish	0.876	0.893	0.957

Table 2. Comparison of CPSNR

Test Images	**CPSNR**		
	Nearest Neighbor	**Bi-Cubic**	**Proposed**
Chick	29.59	31.67	32.97
Dove	24.97	26.76	28.18
Feather	27.86	29.83	30.64
Fish	28.97	29.86	31.23

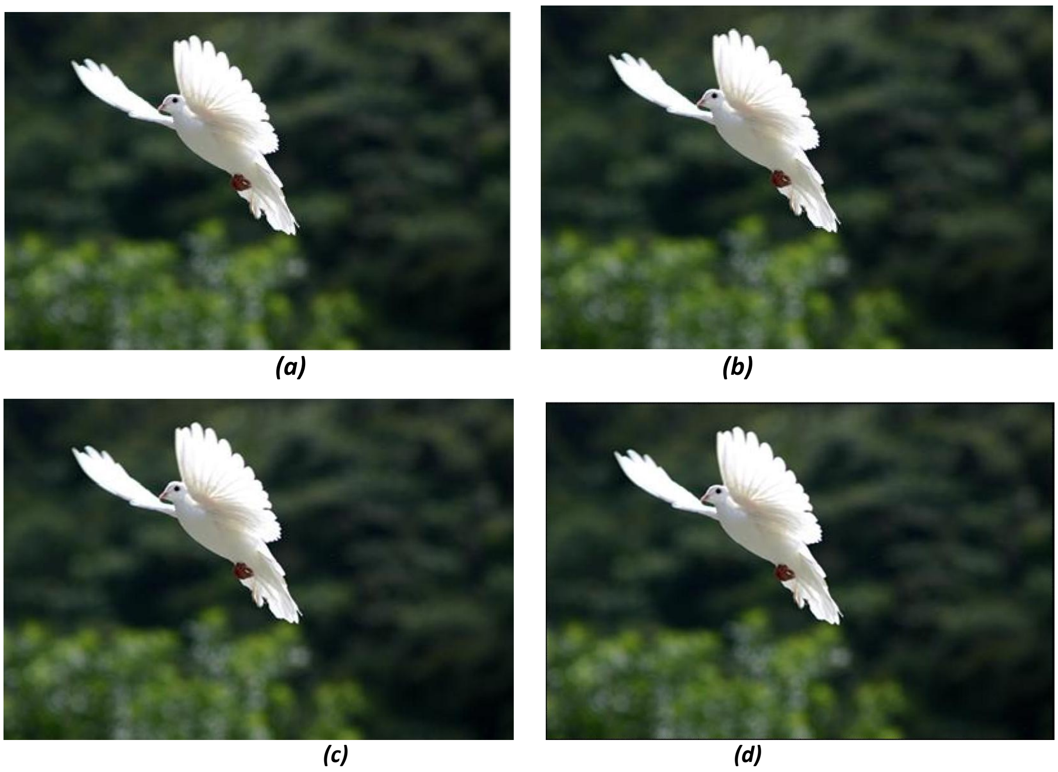

Fig. 4. 2X Zoomed Dove image. (a) Original; (b) Nearest neighbor; (c) Bi-cubic; (d) Proposed

5 Conclusion

The proposed color image interpolation algorithm produces good results in both visual and quantitative analysis. The edges of the images are interpolated along the direction of the edge and hence the interpolation artifacts are greatly reduced in the proposed method compared to other conventional interpolation techniques.

Acknowledgment

I profusely thank my supervisor Dr. R. Amutha, Professor, Department of Electronics and Communication, for her innovative ideas, keen interest, suggestions, guidance and co-operation that paved way for the successful completion of the paper.

References

[1] Lei Zhang and Xiaolin Wu, "An Edge-Guided Image Interpolation Algorithm via Directional Filtering and Data Fusion," *IEEE Transactions on Image Processing,* vol. 15, no. 8, pp. 2226-2238, August 2006.

[2] Hakran Kim, Youngjoon Cha and Seongjai Kim, "Curvature Interpolation Method for Image Zooming," *IEEE Transactions on Image Processing,* vol. 20, no. 7, pp. 1895-1903, July 2011.

[3] Dong-Ho Lee, "A New Edge-based Intra-field Interpolation Method for Deinterlacing Using Locally Adaptive-thresholded Binary Image," *IEEE Transactions on Consumer Electronics,* vol. 54, no. 1, pp. 1110-115, February 2008.

[4] P. Longère, X. Zhang, P. B. Delahunt, and D. H. Brainard, "Pereceptual assessment of demosaicing algorithm performance," *IEEE proceedings*, vol. 90, no.1, pp. 123–132, January 2002.

[5] R. Ramanath, W. E. Snyder, and G. L. Bilbro, "Demosaicking methods for Bayer color arrays," *Journal of Electronic Imaging*, vol. 11, pp. 306–315, July 2002.

[6] R. Kimmel, "Demosaicing: image reconstruction from color CCD samples," *IEEE Transactions on Image Processing*, vol. 8, no. 9, pp. 1221–1228, Sept. 1999.

[7] Xin Li and Michael T. Orchard, "New Edge-Directed Interpolation," *IEEE Transactions on Image Processing,* vol. 10, no. 1, pp. 1521-1527, October 2001.

[8] Zhou Dengwen and Shen Xiaoliu, "An Effective Color Image Interpolation Algorithm," *International Congress on Image and Signal Processing*, Vol. 2, pp. 984-988, 2011.

[9] Jinglun Shi and Zhilong Shan, "Image Interpolation Using a Variation-Based Approach," *IEEE Proceedings*, 2011.