Adaptive Recovery of Image Blocks Using Spline Approach

Jong-Keuk Lee Ji-Hong Kim Jin-Seok Seo

Dongeui University, Busan, Korea

Summary
In this paper, a new method for recovering image blocks is proposed. The proposed method adopts adaptively the spline approach which is used for achieving the smooth curve in computer graphics. For recovering image blocks, the cardinal spline is applied to each damaged block using one dimensional approach at four directions. After calculating selectively the row-wise, column-wise, diagonal -wise, and/or inverse diagonal-wise point functions according to the edge pattern of image area adjacent to the damaged block, the recovery process is performed by determining the weighted sum of selected point functions. By doing so, it can be possible to retain the feature of edges in the recovered image. From the simulation results, it is shown that the proposed method has an excellent performance for all types of damaged blocks compared with conventional approaches.

Key words:
block recovery, cardinal spline, damaged block, image recovery

1. Introduction

The block based image coding is widely used for compressing image and video data. The international standards such as JPEG, MPEG, and H.26x also adopt the block based approach in discrete cosine transform(DCT), motion estimation and compensation(ME/MC), *etc*. The major drawback of the block based coding is to have a block loss in case of occurring errors in channel transmission or decoding process. There are some studies for recovering the block loss. One thing in common of these studies is to utilize the adjacent pixels for estimating the lost block[1][2][3].

In this paper, we propose a new method of recovering the block loss. The proposed method uses a spline approach which is employed for drawing smooth curves in computer graphics. Among a variety of spline approaches, the cardinal spline is employed because it achieves interpolated curve points by using four control points without modifying the value of these points. In the process of recovering the block loss, the edge direction is first found out by using the pixels adjacent to the damaged block and next the cardinal spline is applied to the selected direction. For simple application, the directions of adjacent pixels are considered to be one of four directions such as row-wise, column-wise, diagonal-wise, and inverse diagonal-wise directions. After finding out the direction of adjacent pixels, one dimensional cardinal spline is applied to the selected direction and its inverse direction. The final recovered block is determined by calculating the weighted sum of the results. The weight value is decided by considering the patterns of blocks adjacent to the damaged block. From the simulation results, it is shown that the proposed approach has an excellent performance for all types of damaged blocks.

The organization of this paper is as follows. In section 2, the cardinal spline is briefly reviewed. The proposed method for recovering the damaged block is described in section 3 and the simulation results using the proposed method are presented in section 4. Finally conclusions are drawn in section 5.

Manuscript received February 5, 2011
Manuscript revised February 20, 2011

2. Review of Cardinal Spline

A spline is a flexible strip used for producing a smooth curve through a designated set of points(control points)[4][5]. Among various splines, a cardinal spline is an interpolating piecewise cubic polynomial and completely specified with four consecutive control points, $\mathbf{p}_{k-1}$, $\mathbf{p}_k$, $\mathbf{p}_{k+1}$, and $\mathbf{p}_{k+2}$. The middle two control points, $\mathbf{p}_k$ and $\mathbf{p}_{k+1}$, are the section endpoints, and the other two control points, $\mathbf{p}_{k-1}$ and $\mathbf{p}_{k+2}$, are used for calculating the endpoints slopes. The boundary conditions in the cardinal spline are as follows.

$$\begin{aligned} \mathbf{P}(0) &= \mathbf{p}_k \\ \mathbf{P}(1) &= \mathbf{p}_{k+1} \\ \mathbf{P}'(0) &= 1/2(1-t)(\mathbf{p}_{k+1}-\mathbf{p}_{k-1}) \\ \mathbf{P}'(1) &= 1/2(1-t)(\mathbf{p}_{k+2}-\mathbf{p}_k) \end{aligned} \quad (1)$$

As in Eq. (1), the slopes at control points $\mathbf{p}_k$ and $\mathbf{p}_{k+1}$ are proportional to the slopes $\overline{\mathbf{p}_{k-1}\mathbf{p}_{k+1}}$ and $\overline{\mathbf{p}_k\mathbf{p}_{k+2}}$, respectively. A parametric cubic point function $\mathbf{P}(u)$ for the curve section between control points $\mathbf{p}_k$ and $\mathbf{p}_{k+1}$ is represented by

$$\begin{aligned} \mathbf{P}(u) = {} & \mathbf{p}_{k-1}CAR_0(u)+\mathbf{p}_kCAR_1(u)+ \\ & \mathbf{p}_{k+1}CAR_2(u)+\mathbf{p}_{k+2}CAR_3(u) \end{aligned} \quad (2)$$

where the polynomials $CAR_k(u)$ for k = 0, 1, 2, 3 are

$$\begin{aligned} CAR_0(u) &= -su^3 + 2su^2 - su \\ CAR_1(u) &= (2-s)u^3 + (s-3)u^2 + 1 \\ CAR_2(u) &= (s-2)u^3 + (3-2s)u^2 + su \\ CAR_3(u) &= su^3 - su^2 \end{aligned} \quad (3)$$

and $s = (1-t)/2$ with a tension parameter *t*.

3. Recovery of Block Loss Using Cardinal Spline

The overall procedure of the proposed block recovery method is presented in Fig. 1. The first step of recovering the block loss is to detect the edge direction of the pixels adjacent to the damaged block, and then one dimensional cardinal spline is applied to the detected direction and its inverse direction. In the next step, the weighted sum of the results is determined for achieving final recovered block.

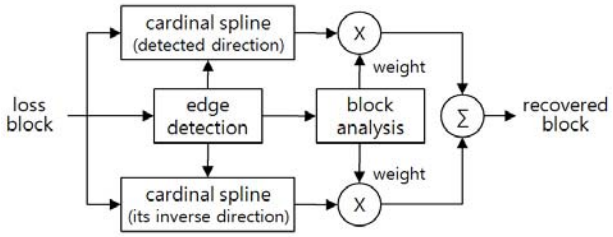


Fig. 1. The overall procedure of the proposed block recovery method

3.1 Edge Direction Detection Using Adjacent Blocks

For detecting the edge direction, Sobel operator is applied to the pixels adjacent to the damaged block as shown in Fig. 2. In Fig. 2, the white boxes with asterisk mark mean the pixels used for detecting the edge direction.

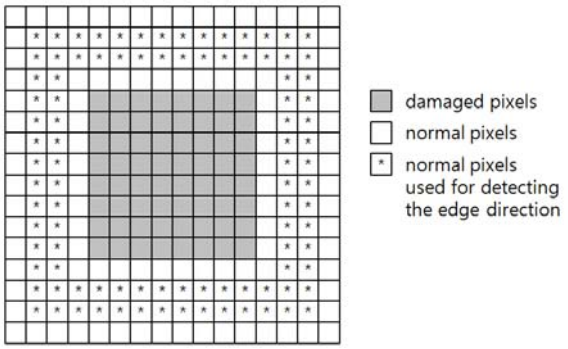
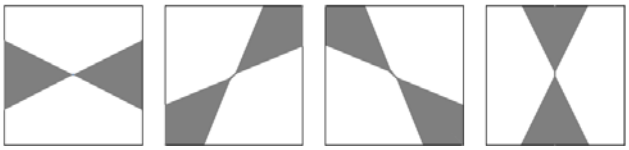


Fig. 2. Edge detection using adjacent pixels

In the proposed method, only four edge directions are considered for simple processing, that is, all edge directions are grouped into four directions 0, 45, -45, and 90 degrees as in Eq. (4) and Fig. 3.

$$angle = \begin{cases} 0^O & \text{if } angle >= -22.5^O \text{ or } angle < 22.5^O \\ 45^O & \text{if } angle >= 22.5^O \text{ or } angle < 67.5^O \\ -45^O & \text{if } angle >= -67.5^O \text{ or } angle < -22.5^O \\ 90^O & \text{if } angle >= 67.5^O \text{ or } angle < 122.5^O \end{cases} \quad (4)$$

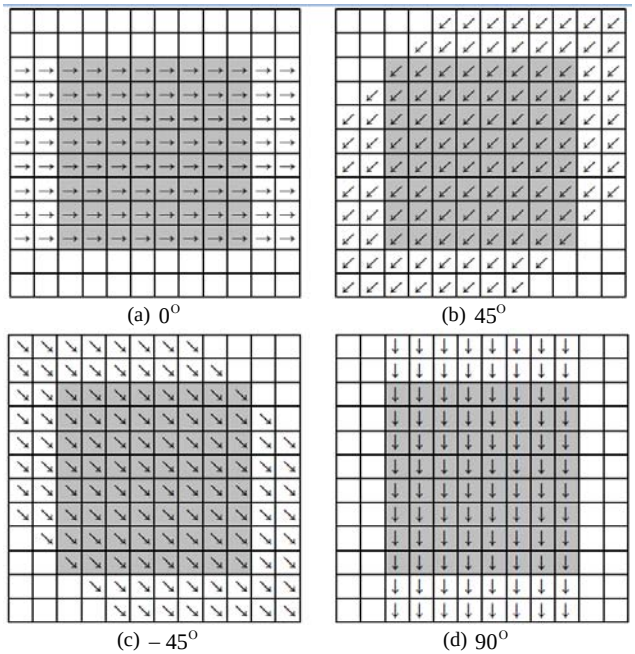


(a) 0^O (b) 45^O (c) -45^O (d) 90^O

Fig. 3. The edge directions grouped into four angles

3.2 Application of Cardinal Spline to 1-D Image Data

After finding out the direction of the blocks neighboring to the damaged block, one dimensional cardinal spline is applied to the selected direction and its inverse direction. For example, the selected direction is row-wise(column-wise), its inverse direction is column-wise(row-wise), and the selected is diagonal-wise(inverse diagonal-wise), its inverse is inverse diagonal-wise(diagonal-wise). In Fig. 4, the application of the cardinal spline for four directions is described, where 0^o, 45^o, -45^o, and 90^o stand for row-wise, diagonal-wise, inverse diagonal-wise and column-wise direction, respectively.



(a) 0^O (b) 45^O (c) -45^O (d) 90^O

Fig. 4. Application of cardinal spline at each direction

Finding the point function $\mathbf{P}(u)$ for row-wise and column-wise directions is performed as follows. In the case of damaged image block of size $N \times N$, the $(N + 4) \times (N + 4)$ block is first selected including the damaged block as in Figure 4. The additional four pixels in each direction is for control points, $\mathbf{p}_{k-1}$,

$\mathbf{p}_k$, $\mathbf{p}_{k+1}$, and $\mathbf{p}_{k+2}$, in the cardinal spline. Applying the cardinal spline for the $(N + 4) \times (N + 4)$ block, one dimensional approach is used. In the row-wise processing, each two pixels in the left and right sides are normal pixels without any damage and middle N pixels are damaged pixels as shown in Figure 5. In applying the cardinal spline, the leftmost and rightmost pixels are designated as control points as $\mathbf{p}_{k-1}$ and $\mathbf{p}_{k+2}$, respectively, and two adjacent pixels are used as control points $\mathbf{p}_k$ and $\mathbf{p}_{k+1}$, respectively. For finding the point function, Eqs. (2) and (3) are applied to the one dimension pixel data and the recovery process is same as finding the value of the point function at parameter $u = 1/(N+1), 2/(N+1), \ldots, N/(N+1)$. For example, the values of the parameter u at each point are 1/9, 2/9, …, 8/9 when $N = 8$. After processing for all row components, the column-wise processing is performed to the original data.

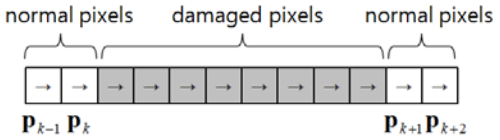


Fig. 5. Pixels in row-wise processing(white: control points used in cardinal spline, gray: pixels to be recovered using cardinal spline)

For the diagonal-wise and inverse diagonal-wise directions, the process of finding the control point is similar to that of row-wise and column-wise directions. But it should be noted that the number of pixels to be processed is variable. The value of parameter u at each direction is described as follows.

오류! 연결이 잘못되었습니다. (5)

where N_d stands for the pixels number to be processed at direction d. For example, N_0 and N_{90} have the value of 8, and N_{45} and N_{-45} have the range of 5 ~ 12 if the damaged block has the size of 8.

3.3 Weighting Outputs of 1-D Processing

After completing processing at one direction and its inverse direction independently, the recovered block is determined by calculating weighted sum of two resulting data. The weight value is decided by considering the magnitude of the edge at the selected direction and its inverse direction. Notating the resulting data at location (x,y) of the selected direction and its inverse direction as $RST_{sel}(x,y)$ and $RST_{inv}(x,y)$, the recovered block data $REC(x,y)$ is

$$REC(x,y) = \alpha \cdot RST_{sel}(x,y) + (1-\alpha) \cdot RST_{inv}(x,y) \quad (6)$$

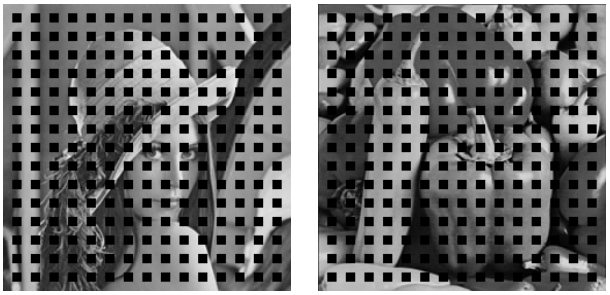
where $\alpha > 0.5$.

4. Simulation Results

In order to test the performance of the proposed method, we simulated our approach on test images. The test images are Lena and Peppers as shown in Figure 6. In the simulation, the damaged test images are artificially made and the damaged blocks are of size 8×8 as in Figure 7. In addition, the value of tension parameter t is fixed to 0.5.

(a) (b)

Fig. 6. Test images (a) Lena (b) Peppers



(a) (b)

Fig. 7. Test images with damaged blocks (a) Lena (b) Peppers

For evaluating the proposed scheme, the conventional approaches of [3] and the neighbor block averaging method are also simulated. Simulation results are represented in Figure 8. In Figure 8, (a, b), (c, d), and (e, f) are the results of using the proposed approach, the method of [3], and the neighbor block averaging method, respectively. As shown in Figure 8, when using the proposed scheme, damaged blocks in all test images are perfectly recovered except at some abrupt transition regions. In Lena, face, hat, and background regions are well recovered, but some damages in abrupt edge region such as shoulder are a little remained. All the damaged blocks in Peppers are perfectly recovered. Using the method of [3], the damaged block especially at the abrupt transition region is remained, especially at abrupt transition regions such as edge. Only some damaged blocks at flat regions are recovered to a certain degree. When using the method of the neighbor block averaging method, nearly all the damaged blocks except at the flat region are remained. For more detailed comparison, the magnified images at some region are presented in Fig. 9. The magnified regions are face of Lena and stalk end of Peppers. As shown in Fig. 9, the damaged blocks are nearly recovered when using the

proposed method, but there remain unrecovered the damaged blocks in the conventional methods. From the simulation results, it can be seen that the proposed method has excellent performance of recovering the damaged blocks compared to the conventional methods.

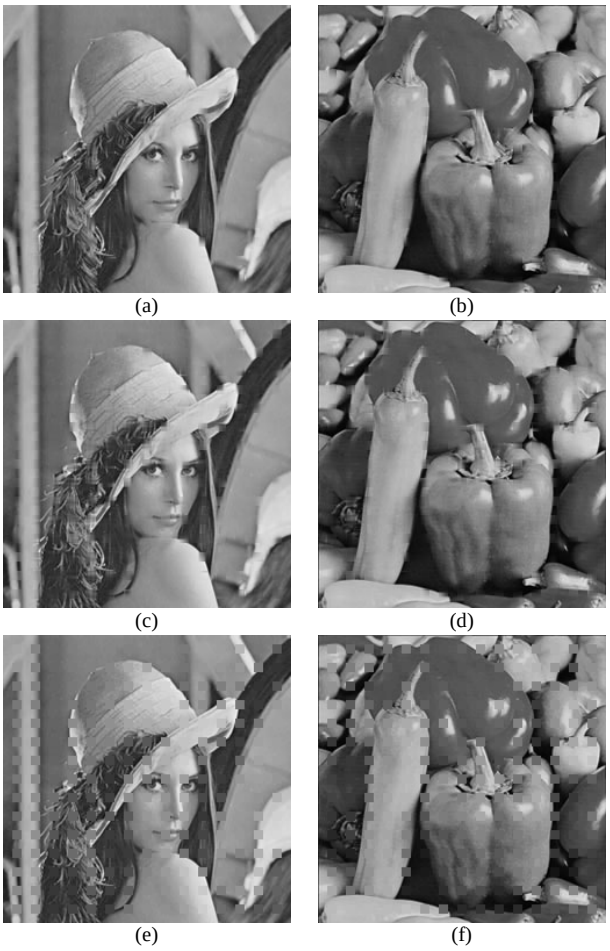


Fig. 8. Simulation results. (a, b): the proposed method, (c, d): the method of [3], (e, f): the neighbor block averaging method

5. Conclusions

In this paper, a new scheme for recovering the damaged blocks was proposed. The proposed method uses the spline approach that is utilized in computer graphics for achieving the smooth curve. The proposed approach has the major merit of greatly recovering the damaged blocks for nearly all the types of damage patterns. The key point of the proposed approach is to estimate the edge direction of the damaged blocks by using neighboring normal pixels. In addition, by adopting the cardinal spline, it can be possible to use normal pixels adjacent to the damaged block as control points.

From the simulation results, it was shown that the proposed scheme has excellent performance for all types of damaged patterns.

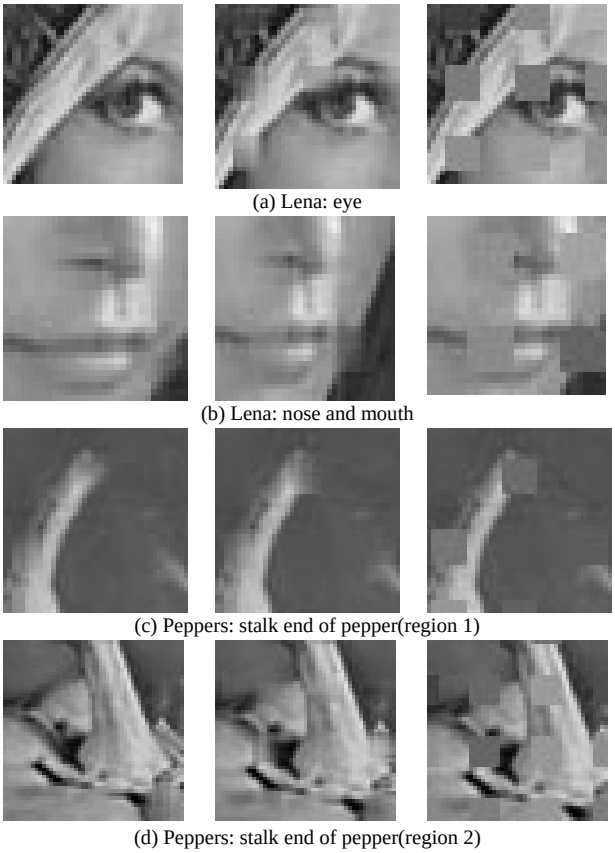


Fig. 9. Magnified images. left: the proposed method, center: the method of [3], right: the neighbor block averaging method

Acknowledgments

This work was supported by Dong-eui University Foundation Grant (2009)
This work was supported by the Culture Technology Joint Research Center Project of the Korea Creative Content Agency

References

[1] Y. Wang and Q. F. Zhu, "Error Control and Concealment for Video Communication: A Review," Proc. IEEE Multimedia Signal Processing, May 1998.
[2] J. P. Park, D. C. Park, R. J. Marks, and M. A. El-Sharkawi, "Recovery of Image Blocks Using the Method of Alternating Projections," IEEE Transactions on Image Processing, Vol. 14, No. 4, April 2005.
[3] J. H. Kim and J. S. Seo, "Recovery of Damaged Image Blocks Using Cardinal Splines," Journal of Electronics and Computer Sciences, Vol. 12, No. 2, December 2010.
[4] D. Hearn and M. P. Baker, Computer Graphics C-version 2nd Ed., Prentice Hall, 1996.

[5] J. D. Foley, C. V. Dam, S. K. Feiner, J. F. Hughes and R. L. Phillips, Introduction to Computer Graphics, Addison-Wesley Publishing Co. Inc., 1994.
[6] Lee Heung Kyu, Digital Image Processing-Theory and Implementation, SciTech Media, 2007.
[7] Y. Yang and N. P. Galatsanos, "Removal of Compression Artifacts Using Projections onto Convex Sets and Line Processing Modelling," IEEE Transactions on Image Processing, Vol. 10, No. 10, October 1997.

Jong-Keuk Lee received the B.S. degree in electronics engineering from Kyungpook National University, Korea, in 1978, the M.S. and Ph.D. degrees in computer science from North Carolina St. University and Texas A&M University in 1988 and 1994, respectively. He is now a professor at Dong-eui University, Busan, Korea. His current research interests are in the area of parallel processing system, distributed system and ubiquitous computing.

Ji-Hong Kim received the B.S. and M.S. degrees in electronics engineering from Kyungpook National University, Korea, in 1986 and 1988, respectively, and the Ph.D. degree in electronic and electrical engineering from POSTECH, Korea, in 1996. He worked at ETRI from 1988-1997 and at Pusan University of Foreign Studies form 1997-2002. In 2002, he joined the Dong-eui University, where he is an associate professor. His current research interests are in the area of digital image processing, computer graphics, and computer vision.

Jinseok Seo received the B.S. degree in computer engineering from Konkuk University, Korea, in 1998, and the M.S. and Ph.D. degrees in computer science and engineering from POSTECH, Korea, in 2000 and 2005, respectively. He is now an assistant professor at Dong-eui University, Busan Korea. His current research interests are in the area of computer game, authoring tool, virtual reality and augmented reality.