The Image Shape Recovery on Communication Data Lost Damage

Shih-Chang Hsia, Ming-Hwa Sheu /National Yunlin University of Science and Technology, Department of Elecrtron, Douliou, Taiwan
Email: hsia@yuntech.edu.tw

Cheng Hung Hsiao / GeoVision Inc, Taipei, Taiwan
Email: evanse@gmail.com

Abstract**—**This paper presents a fast efficient error concealment method for recovering information related to shape. The proposed technique comprises block classification, edge direction interpolation and filtering interpolation. Missing blocks are classified in four categories: transparent, opaque, edge and isolated blocks. Most of the computation is spent on edge blocks and isolated blocks to maximize the cost and performance tradeoff. For the recovery of edge blocks, the edge slope is computed by referring to the nearest available block, from which the missing shape is interpolated parallel to the edge. Isolated blocks are dealt with using a cascade filter to approximate the actual shape**..**

Keywords—GSM/GPRS, watermeter, wireless network

I .Introduction

There are two general approaches to overcoming shape error: spatial processing and temporal processing [1-5]. In this paper, we study spatial interpolation to recover missing intra-shape blocks. Simple bilinear interpolation can reconstruct shape information using neighboring data. This approach is generally adopted for computational simplicity rather than performance. Edge-directed interpolation was employed by Ma et. al. [6], which estimated based on the edges in the neighboring frames. Unfortunately, the computational load of this approach is very high and performance is strongly dependent on image features. Shirani et. al. presented a novel error-concealment method for recovering shape information [7], which iteratively interpolates missing pixels using weights in eight directions. Iterative procedures enable the gradual recovery of shape error; however, the recovered shape does not appear smooth in the final results.

This paper presents a new approach to the recovery of object shape for the intra-frame and still images. The proposed algorithm comprises a block classification scheme in conjunction with edge interpolation to improve quality without increasing computational costs.

II. Proposed Algorithm

Receiving a decoded frame with an error message from the previous decoding stage enables the location of missing block to be identified. Processed blocks can be classified into four types, with the features of neighboring blocks, as shown

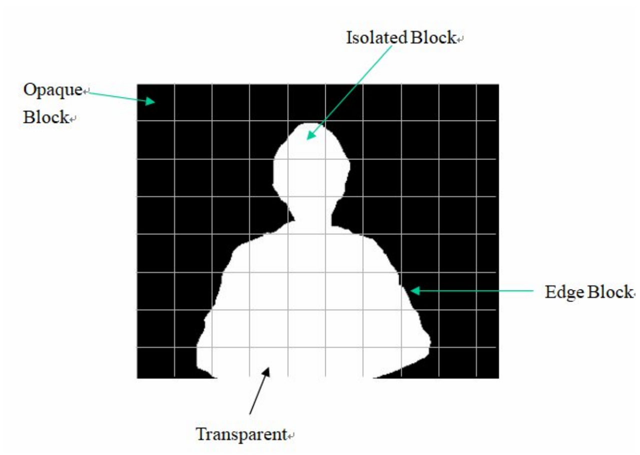

Fig. 2 The block classification for the binary shape plane

in Fig. 1. To improve performance, each type of block is processed independently using different methods. If neighboring blocks are all dark or all bright, the current-block type will be either opaque or transparent. The kind of block is easily recovered because the data can simply be replicated using adjacent pixels. Edge blocks provide shape data that is partially dark or partially bright because the neighboring blocks contain edge information. Edge direction can be determined using the top and bottom blocks to interpolate lost shapes with a high degree of accuracy. Isolated blocks can be classified with neighboring blocks of various types. For example, the top and bottom blocks adjacent to the isolated block in Fig. 1 are opaque and transparent, respectively. This kind of block is always located at the top or bottom of an object; therefore, the spatial correlation between the processed block and neighboring blocks is quite low. Isolated blocks are extremely difficult to recover since they lack the information required to restore the edge of the shape.

For opaque and transparent block, they are directly copied form neighboring blocks. We mainly discus the edge block and isolated block. we describe the method of processing edge blocks. The XOR operation is used to estimate the edge

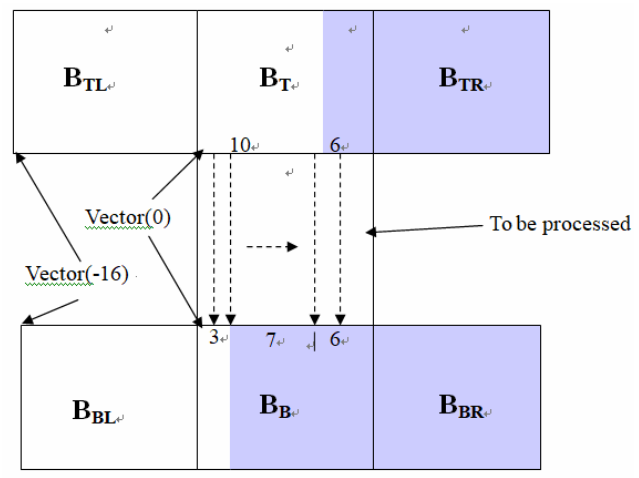

Fig. 2(a) The estimation of edge direction for the missing block

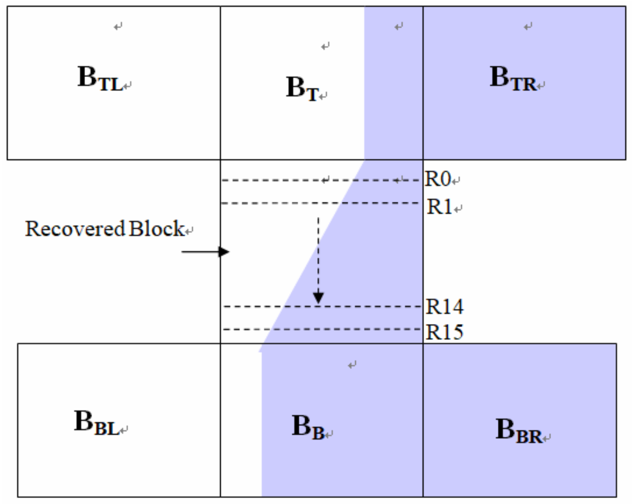

Fig. 2(b) Interpolation along the edge direction**.**

direction. If the block size is N×N, the XOR function of the T-block and B block is performed as follows:

$$T\_\{Xor\} = (\underline{B_{TL}^{0} \oplus B_{TL}^{1}}, \underline{B_{TL}^{1} \oplus B_{TL}^{2}} ..., \underline{\oplus B_{TL}^{N-1} \oplus B_{T}^{0}}, \underline{B_{T}^{0} \oplus B_{T}^{1}} .., \underline{\oplus B_{T}^{N-1} \oplus B_{TR}^{0}}, ... \oplus B_{TR}^{N-1}) ,$$

$$B\_\{Xor\} = (\underline{B_{BL}^{0} \oplus B_{BL}^{1}}, \underline{B_{BL}^{1} \oplus B_{BL}^{2}} ..., \underline{\oplus B_{BL}^{N-1} \oplus B_{B}^{0}}, \underline{B_{B}^{0} \oplus B_{B}^{1}} .., \underline{\oplus B_{B}^{N-1} \oplus B_{BR}^{0}}, ... \oplus B_{BR}^{N-1}) \quad .$$

(1)

Where the symbol ⊕ is for XOR. The XOR function is performed for each of adjacent pixels. For example, in Fig. 2(a), *T_{Xor}*=(0,..,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,..,0) and *B_{Xor}*=(0,..,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,..,0), where the symbol "0,..,0" includes 16 zeros. The output of (2) is a set of 47 logical values for checking the boundary vector for one 16x16 block processing. We define that vector (0) as being located at the first pixel of block B_T and B_B, and the first pixel of B_{TL} and B_{BL} will be vector (-16). Thus, the vectors of block B_T and B_B are 10 and 3, respectively. The edge direction is between the two vectors. If the adjacent top and bottom blocks adjacent to the missing block have the same edge direction, clearly the edge of the lost block is vertical and the T-blocks and B-blocks will have the same vector value. If the distance to the boundary vector of the T-block is greater than that of the B-block, the edge slope will be between 0^0 and 90^0. Otherwise, it will be between 90^0 and 180^0.

The missing block is then interpolated along the edge direction of the T- and B-blocks. First, we can interpolate the edge line between the vector of the T- and B-blocks, where the result is shown in Figure 2(b). Because shape information is binary, we can draw an edge line using 1 between the vector (10) of the T-block and the vector (3) of the B-block. Next, the other missing pixels can also be filled in using 0 or 1 by referring to the edge line. In this case, the missing pixels are interpolated using 1 from the first pixel of each row to the edge line, and are filled using 0s from the edge line to the last pixel of each row. The edge line of the jth row can be determined for the lost block using two cases provided by

$$\begin{cases} if\ V_T > V_B, R_j = \mathrm{V_T} - \mathrm{int}((\mathrm{V_T} - \mathrm{V_B})/16) \times (\mathrm{j}+1) \\ if\ V_T < V_B, R_j = \mathrm{V_T} + \mathrm{int}((\mathrm{V_B} - \mathrm{V_T})/16) \times (\mathrm{j}+1) \end{cases} , \qquad (2)$$

where the size of the lost block (N) is 16×16, and symbols V_T and V_B denote the vector of the top and bottom blocks, respectively. R_j is the location of the edge point in the jth row, and symbol "*int(b)*" takes the integer value of b. For example, the 14th row edge, R14=3+*int*((10-3)/16)×15=9 in Fig. 2(b).

As mentioned above, the proposed algorithm is used for vertical slope-edge blocks. However, the object has horizontal edges, which are difficult to recover using spatial processing because the information related to horizontal edges is lost when slice data is corrupted. Horizontal edge blocks are classified as isolated blocks because the adjacent top and bottom blocks belong to different block types.

Next, we turn our attention to the processing of isolated blocks. The shape information in Cases 1~6 cannot be recovered efficiently using the edge information of the T- and B-blocks. We propose the use of a local approximate interpolation filter to recover the isolated block efficiently. We first perform linear interpolation for the lost block with boundary pixels of the B_T and B_B blocks. As *N=16*, the *i*th pixel of the first vertical line can be interpolated using

$$\hat{f}_{i,o}^{B_M} = f_{0,0}^{B_B} \times \frac{i}{16} + f_{N-1,0}^{B_T} \times \frac{16-i}{16} \text{ , } i=0 \text{ to } 15 \qquad (3)$$

where $\hat{f}_{i,0}^{B_M}$ is the interpolated result of the first vertical line for the missing block B_M, $f_{0,0}^{B_B}$ is the first row and first column of the bottom block B_B ,and $f_{N-1,0}^{B_T}$ is the last row and first column of the top block B_T, as shown in Fig. 3(a). The interpolation result from Eq. (3) will be a gray level because the top block B_T is bright with a value of 255 in an 8-bit image and the value of the bottom block B_B is 0. According to (4), we can interpolate all vertical lines of the lost block with a column-by-column approach. After interpolation, the missing block appears as a gray level because the levels of the boundary of the top and bottom block of the isolated block are inversed.

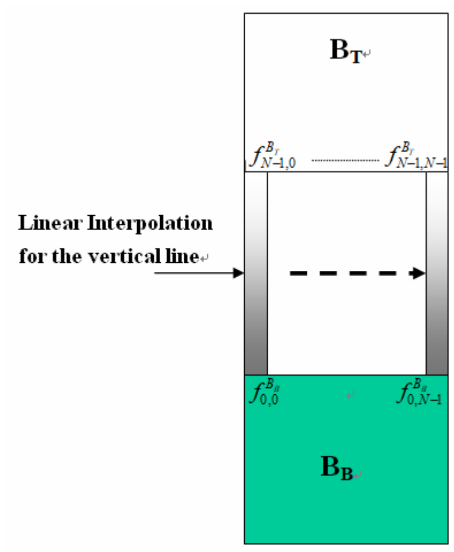

Fig. 3(a) The linear interpolation for the corrupted block

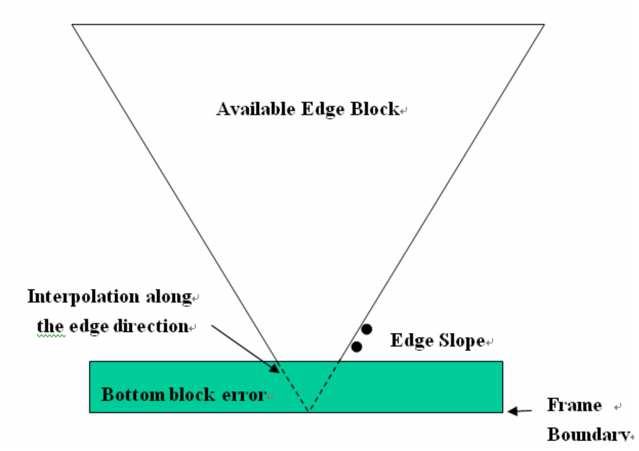

Fig. 3(b) Interpolation for bottom block error along the edge direction of T-block.

The interpolated value is further processed using a low-pass filter to recover the error blocks as approximate real shapes using the approximation approach. The low-pass filter is only used for the error pixels of isolated blocks. To avoid violating our objective, the other block types and the normal pixels are not processed using the low-pass filter. The reason is that we wish only to use the low-pass filter to spread the information of normal shapes in neighboring blocks to modify the gray level of the error blocks. The low-pass filter was designed using an N×M window, which may cover pixels of the true shape from the blocks adjacent to the boundary. Following recursive low-pass filtering, the gray-level for error pixels is modified to nearly match the shape, which becomes increasingly close to the real shape with an increase in the number of processing stages.

For example, if the processing window covers object pixels with a level of 255, the gray level of the error pixel after low-pass filtering will increase. Closer to the background, the level will decrease since the window maybe cover more black pixels. This approach can be used to efficiently modify the bilinear interpolation results for the lost block when moving toward the actual shape of the object.

Considering complexity and quality tradeoffs, this study designed a simple, multiplierless low-pass filter using a 3×5 window to reduce computation with a cascading structure, which requires only three line-buffers. We employed a 4-bit right-shifter rather than one division to reduce computational cost. Using pixel by pixel processing, the gray level of the lost block trends to the shape outline because the recursive low-pass filter spreads available data from neighboring blocks to the lost block. The recursive low-pass filter uses a cascade structure, such that the results of the first low-pass filter are processed using subsequent low-pass filters. Finally, the gray-level is truncated at a threshold to form binary shape data. The threshold value is determined according to the number of available neighboring shape blocks, denoted by P. The threshold value is determined by 128-8×(6-P). If the processed block is close to the object, the threshold will be higher, in which case all neighboring blocks are available and P=6. When two-slice error appears in continuous blocks in vertical direction, the threshold is decreased to encompass the information of the entire shape, where P is reduced.

Figure 3(b) illustrates the error concealment using edge prediction for the bottom block along the edge slope of the shape. We employed a 5×9 window to predict the direction of the shape. Figure 3(c) shows the processing window for the error concealment in the bottom blocks, in which the "dash block" is processed. The first three rows are the available shape, and the last two rows need to be interpolated. First, we search the shape edge located in the last row of the T-blocks using the center point of the window (j=2, i=4), and then check the shape edge in the 1st row (j=0) and the 2nd row (j=1). The shape edge in the 3rd and 4th rows is restored from (4-j,8-i) as the shape location of the 1st and 2nd rows. For example, the error concealment for the shape location in the 4th row at (4, 2) is found the shape of the 1st row is at (j=0,i=6) in Figure 6(c). In the same manner, shape recovery for the 3rd row (j=3) is estimated using the first row (j=1). The recovered shape in the 3rd row at (3,3) is found at (1,5). When an edge dot is found, we can fill "1" from the first column to the edge dot, and interpolate "0" from the edge dot to the last column. When the two rows are completely recovered, the window moves to the next row and uses the same procedures to interpolate the missing shape to the boundary. This approach enables the correction of the most shape information for the bottom block. By the same manner, we can recover the shape of the lost block at the top slice.

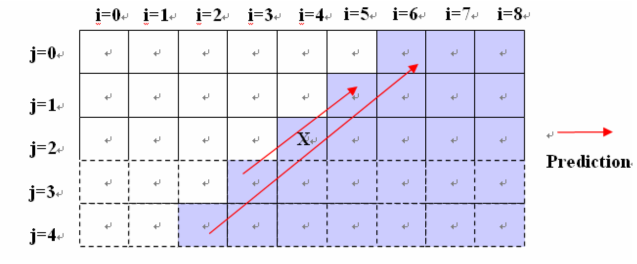

Fig. 3(c) Prediction for the shape with a 5× 9 window

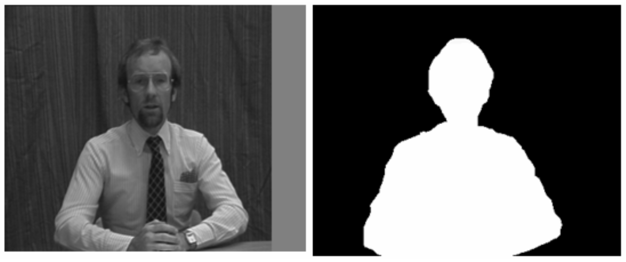

Fig. 4 The original picture "TV" and its shape.

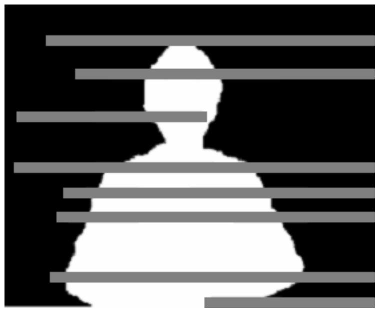

Fig. 5 26% image lost.

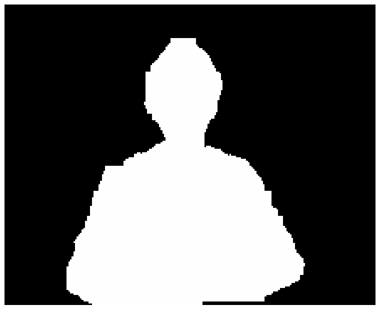

Fig 6 The error recovery with bilinear

3. Simulations and Comparisons

In the following, we evaluate the performance of the proposed algorithm. First, three video sequences were selected,

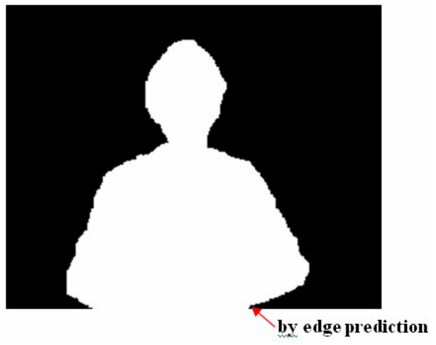

Fig. 7 The shape recovery with the proposed method.

and shape information of each object was extracted, as shown in Figures 4. The frame size employed the CIF (Common Intermediate Format) with 352×288. The videos were encoded by referring to the shape information. The coded bit-stream will be transmitted to the decoder through an error-prone channel. Errors were generated randomly using a random generator function. Since "Slice" is a minimum synchronous unit in video coding, which introduces the possibly of partial or complete slice errors. The block size used was 16×16, which accounts for 16×16×m of corrupted data, where m is the number of error blocks. For simulations of shape error concealment, 16×16×m data is corrupted by random pattern in image blocks. The shape images in the first frame are corrupted as shown in Figs. 4, in which the error rate is about 26% of the macroblocks lost. We compared the performance of the proposed system with pixel-level simulations using bilinear interpolation [1], and the fuzzy theory method [5] Each corrupted image was recovered with each of the methods. We first employed bilinear interpolation to fill the missing block and then truncated the interpolated results as a threshold of 127. The results are shown in Figs. 6. The performance is poor, as indicated by the lack of smoothness resulting from distortion in the shape outline. Fuzzy theory was used to refine results of bilinear processing, which were also truncated to obtain binary shape information. The results from the Fuzzy method were better than those obtained using the bilinear method. Results can be evaluated using an error point that represents the difference in the number of shape pixels between the original and reconstructed alpha planes. The error points can be checked by an XOR operation of the original and restored images. If the pixels in the restored image differ from pixels in the same position of the original image, the XOR operation returns a high value. When all pixels in a frame are computed using XOR, the high values are accumulated to denote the number of error points.

In the following, we present the results of our proposed algorithm. The shape error image first is processed using bilinear interpolation for isolated blocks. The results were further processed using local filtering approximation through several stages of low-pass processing. As the number of low-pass stages increased, the gray-level of the error block grew increasingly closer to the real shape. Finally, the gray-level was truncated to binary with an adaptive threshold. We found that the results were satisfactory following three stages of low-pass cascaded filtering. In fact, this approach efficiently recovered the shape error not only for isolated blocks but also for edge blocks. To reduce computation time, we employed a simple logic operation to recover the edge block. The complete results are shown in Figs. 7 for error shape recovery, in which a 3-stage low-pass filter is used for isolated blocks. The proposed method can efficiently restore shape information, outperforming the bilinear and fuzzy approaches. In a special case, the missing block at the bottom is simulated with Fig.5. Clearly the B-blocks are not available. However, two-boundary points are required for the bilinear interpolation. In simulations, the non-available pixels use 0s for bilinear computation and the results from other forms of interpolation are poor. The proposed algorithm recovers the shape information for the missing block at the bottom by the vector of the edge direction and shape error can be efficiently concealed by edge prediction, as shown in Fig. 7.

V. Conclusions

This paper presents a fast-efficiency method for the recovery of shape information. First, the processed blocks are classified into four categories. We then employ various techniques to process each type of block to restore the shape information and improve performance. Following block classification, no computational power needs to be spent processing opaque or transparent blocks since these blocks can be easily restored simply by copying neighboring pixels. Much more computation is dedicated to edge blocks and isolated blocks because the quality of these blocks largely determines recovery performance. To restore edge blocks, we propose an edge-oriented algorithm using the boundary check method with simple logic operations rather than complex computation. Spatial interpolation is used to produce the pixel for the lost block parallel to the edge. Error concealment for the isolated block is the most difficult because it lacks available blocks in one direction. The filtering approximation approach is used to recover shape information for isolated blocks.

References

1.Sohel, F., Bennamoun, M., Hahn, M.," Spatial shape error concealment utilizing image texture," *IEEE Conf. Industrial Electronics and Applications,* 2011, pp. 265-270.
2.Uhlíková, I., Benešová, W., Polec, J., Csóka, T., "Texture aware image error concealment," *IEEE EUROCON Conf. Computer as a Tool*, 2015, pp.1-6.
3.Dun,Y., Liu, G.,"An extrapolation method for MDCT domain frame loss concealment", *IEEE Conf. Signal and Information Processing*, 2014, pp.650 - 654.
4.Ranjbari, M., Sali, A. Karim, H. A. Hashim, F.," Depth error concealment based on decision making," *IEEE Conf. Signal and Image Processing Applications*, 2013, pp.193-196.
5. Lee, P. J., Chen, L. G., Wang, W.J., Chen, M.J.," Robust error concealment algorithm for MPEG-4 with the aids of fuzzy theory", *IEEE Conf. Consumer Electronics*, 2001, PP. 154-155.
6.Ma, M., Au, O., Chan, S.-H. G., Sun, M.T. "Edge-directed error concealment," *IEEE Trans. Circuits and Systems for Video Technology,*. vol. 20, no. 3, Mar. 2010, pp.382-395.
7.Shirani,S., Erol, B., Kossentini, F., "A concealment methods for shape information in MPEG-4 coded video sequences", *IEEE Trans. Multimedia,* vol.2 , no.3, pp.185-190, Sept., 2000.
8.Tsiligianni, E., . Kondi, L. P., Katsaggelos, A. K., "Shape error concealment based on a shape preserving boundary approximation ," *IEEE Conf. Image Processing*, 2010, pp.457-460.