AN INTRA-FRAME ERROR CONCEALMENT: NON-LINEAR PATTERN ALIGNMENT AND DIRECTIONAL INTERPOLATION

W. Kumwilaisak † and F. Hartung‡

†Telecommunication Department, King Mongkut University of Technology, Thonburi, Bangkok, Thailand
‡ Ericsson Research, Mobility Applications Laboratory, Herzogenrath, Germany
E-mail: kumwilai@hotmail.com, Frank.Hartung@eed.ericsson.se

ABSTRACT

This paper presents a new intra-frame error concealment algorithm designed for both image and video with non-linear pattern alignment and directional interpolation. By this intra-frame error concealment algorithm, first the non-linear pattern alignment based on a variant version of the dynamic programming algorithm is applied to image pixels around a corrupted region. The non-linear pattern alignment effectively provides the best aligned pixel pairs from boundary pixels around the corrupted region to dynamically capture local image textures in terms of optimized metric. Then, with the aligned pixel pairs, the directional interpolation algorithm is used to recover image pixels in the corrupted region. From computer simulation in both image and video, the proposed intra-frame error concealment provides much better quality improvement of recovered image pixels comparing to that of the bilinear interpolation approach.

1. INTRODUCTION

Corruption of image and video information, when it goes through the hostile environment, can severely affect the visual quality of reconstructed image and video at the receiver. The distortion of visual quality can be even more serious in case of video, if the corruption occurs in an intra frame. This is because an intra frame is used for predicting the following video frames in a DPCM-based video coding. Therefore, the effective error concealment algorithm for intra frame should be specially considered due to its contribution in significantly improving reconstructed video quality (i.e., reduce the error propagation from intra frame to the following video frames).

There have been some researches studying the intra-frame error concealment algorithm [2, 3, 5]. The principle of their works is based on the modeling of texture representation in the corrupted region by geometric analysis, and then performs the directional interpolation to recover corrupted pixels. However, when there are more than one edge in the corrupted region or not all neighboring blocks are successfully received, the previous proposed algorithms may not effectively derive textures in the corrupted region well. In this paper, we propose a new way to perform the intra-frame error concealment, which can deal with multiple textures in the corrupted region. The proposed approach is based on non-linear pattern alignment and directional interpolation. First the non-linear pattern alignment based on a variant version of dynamic programming algorithm is applied to image pixels around the corrupted region. The non-linear pattern alignment effectively provides the best aligned pixel pairs from boundary pixels of the corrupted region to dynamically capture local image textures in terms of optimized metric. Then, with aligned pixel pairs, the directional interpolation algorithm is used to recover pixels in the underlying corrupted region.

The rest of the paper is organized as follows. Section 2 describes the non-linear pattern alignment algorithm, which searches for the optimal texture representation of the corrupted region. An example of the non-linear pattern alignment algorithm also provides in Section 2. In Section 3, the directional interpolation used to recover corrupted pixels is presented. Simulation results are given in Section 4. Finally, concluding remarks are in Section 5.

2. NON-LINEAR PATTERN ALIGNMENT ALGORITHM

2.1. Problem Formulation

Assume we have two image sequences obtained from image pixels around the corrupted region, which have been successfully received. Define image sequence one as $S_1 = M_1, M_2, \ldots, M_{N_1}$ and image sequence two as $S_2 = P_1, P_2, \ldots, P_{N_2}$, where N_1 and N_2 are the length of S_1 and S_2, respectively, and are not necessary to be equal. M_i and P_i represent the ith pixel of S_1 and S_2, respectively. The example of S_1 and S_2 can be seen in Fig. 1, where S_1 is the pixel row immediately above the corrupted region, whereas

0-7803-8554-3/04/$20.00 ©2004 IEEE.

Fig. 1. The texture representation obtained from the nonlinear pattern alignment algorithm applied to "Lena" image.

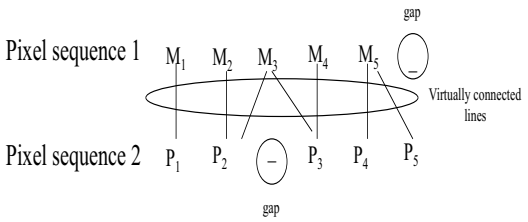

Fig. 2. The alignment of image pixels.

S_2 is the pixel row immediately below the corrupted region.

Our problem will be to find the best texture representation of the corrupted region from image information obtained from S_1 and S_2. To obtain the texture representation, we try to align pixels in two image sequences, S_1 and S_2, by inserting some gaps at different pixel locations of S_1 and S_2. Then, we draw virtually connected lines among the aligned pixels. The set of these connected lines will form the underlying texture representation. The example of the pixel alignment and the set of connected lines between two image sequences are in Fig. 2.

Suppose that $\boldsymbol{\Upsilon}(\boldsymbol{\Omega})$ is a cost function, which is a function of the texture representation Ω. Therefore, our formulated problem is to find the best texture representation Ω^* such that

$$\boldsymbol{\Upsilon}(\boldsymbol{\Omega}^*) = \max_{\Omega} \boldsymbol{\Upsilon}(\boldsymbol{\Omega}), \tag{1}$$

where Ω^* is the texture representation maximizing the cost function. In this paper, we use the dynamic programming approach to solve (1) and obtain the best texture representation.

2.2. Dynamic Programming Approach

In this section, we describe a variant version of the dynamic programming algorithm, which will be used to obtain the best image pixel alignment. There are three possible ways to go, when performing image pixel alignment: inserting a gap to image sequence S_1, inserting a gap to image sequence S_2, and trying for matching between pixels in S_1 and S_2.

We first set up a scoring matrix with dimension $(N_1 + 1) \times (N_2 + 1)$. The columns of the matrix represent the pixels in image sequence S_1, whereas the rows of the matrix represent the pixels in image sequence S_2. This matrix will be used to compute a cost function in searching for the best pixel alignment. The pixel alignment algorithm based on the dynamic programming algorithm can be described step by step as follows.

- Step 1: Fill the first row and the first column of the scoring matrix with zeros.
- Step 2: Fill every position in the scoring matrix by considering as follows. If the absolute difference of pixels in position i of S_1 and position j of S_2 is less than the specific threshold, in other words,

$$|M_i - P_j| < T_{th}, \tag{2}$$

where T_{th} is the underlying specific threshold. Then, the accumulated score of the scoring matrix at position (i, j) is computed from

$$\begin{aligned}\Upsilon(i,j) = \max(&\Upsilon(i-1,j-1)+1,\\ &\Upsilon(i-1,j)+g, \Upsilon(i,j-1)+g),\end{aligned} \tag{3}$$

where $\Upsilon(i, j)$ is the accumulated score of the scoring matrix at position (i, j), and g is a gap penalty when inserting a gap to a image sequence. Note that $\Upsilon(i-1, j-1)+1$ in (3) is the accumulated score resulting from the exact match of pixels at position i of S_1 and at position j of S_2. $\Upsilon(i-1, j)+g$ and $\Upsilon(i, j-1)+g$ are the accumulated scores resulting from inserting a gap in image sequence S_1 and S_2, respectively. In this paper, we assign g to be equal to zero. If (2) is not satisfied, then the accumulated score of the scoring matrix at position (i, j) is computed from

$$\begin{aligned}\Upsilon(i,j) = \max(&\Upsilon(i-1,j-1),\\ &\Upsilon(i-1,j)+g, \Upsilon(i,j-1)+g).\end{aligned} \tag{4}$$

- Step 3: Trace back from the right-bottom corner, which corresponds to the maximum value of accumulated score, to the left-upper corner of the scoring matrix. The direction in tracing back is depended on the derivation of the accumulated score in Step 2. For example, when we trace back from position (i, j), if the accumulated score of position (i, j), is derived from $\Upsilon(i-1, j-1)+1$ or $\Upsilon(i-1, j-1)$, then the direction in tracing back is from position (i, j) to position $(i-1, j-1)$ of the scoring matrix. In contrast, if the accumulated score of position (i, j), is derived from $\Upsilon(i-1, j)+g$ or $\Upsilon(i, j-1)+g$, then the direction in tracing back will be from (i, j) to $(i-1, j)$ or $(i, j-1)$, respectively.

With the tracing back step, we can derive the best alignment of two image sequences. The example of the dynamic programming algorithm for pixel alignment can be shown in Fig. 3. The best pixel alignment obtained from the example in Fig. 3 is shown in Fig. 4. Note that the complexity of the

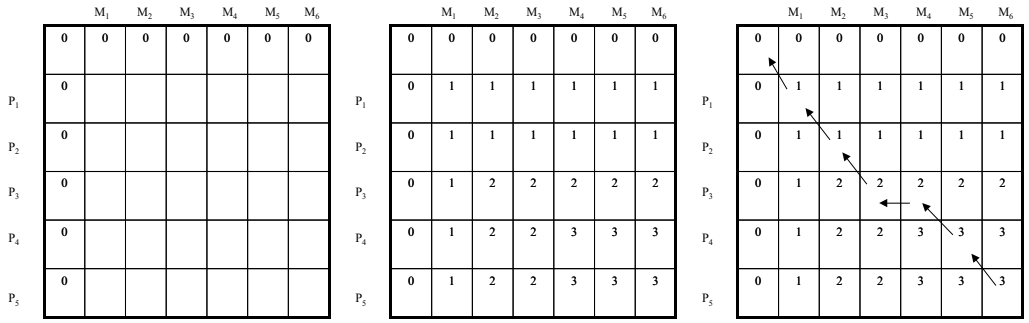

Fig. 3. The example of image pixel alignment based on dynamic programming approach is illustrated. In this example, we assume that $M_1 = P_1 = P_2$, $M_2 = M_3 = P_3$, $M_4 = M_5 = P_4$, and $M_6 = P_5$. The left scoring matrix shows the initialization of the dynamic programming algorithm, where the first row and the first column are filled with zeros as described in Step 1. The middle scoring matrix shows the scoring matrix after filling by algorithm described in Step 2. Note that the directions corresponding to derived accumulated scores are recorded and will be used in the tracing back step. The right scoring matrix shows the tracing back step from the right-bottom position to the left-top position as described in Step 3.

dynamic programming algorithm for image sequence alignment is $O(N_1 \times N_2)$. The alignment of image sequences will be the element in deriving the image texture representation and be used to recover corrupted pixels in the next section.

3. DIRECTIONAL INTERPOLATION

Define the best set of texture representation obtained from the aligned image pixels as

$$\Omega^* = \{(X_1^*, Y_1^*), (X_2^*, Y_2^*), \ldots, (X_k^*, Y_k^*)\}, \qquad (5)$$

where k is a number of aligned pixel pairs of Ω^*, X_k^* and Y_k^* are the pixel values of pixel pair k from image sequence one and two, respectively. Now, from each aligned pixel pair of Ω^*, we draw a virtual line connected between the first element and the second element of pixel pair. For example, with the pixel pair kth, the line is virtually draw between X_k^* and Y_k^*. We define the virtually connected line linked between X_k^* and Y_k^* as l_k. Therefore, there are totally k lines obtained from the aligned pixel pairs in Ω^*. In the example shown in Fig. 3 and 4, the best texture representation $\Omega^* = \{(M_1, P_1), (M_2, P_2), (M_3, P_3), (M_4, P_3), (M_4, P_4), (M_5, P_4), (M_6, P_5)\}$.

With all virtually connected lines, the recovery of corrupted pixels in the corrupted region can be considered into two cases depended on their locations. In the first case, the position of the corrupted pixel is on the virtually connected line. The corrupted pixel can be recovered from the aligned pixel, which is used to draw the virtually connected line. For example, in Fig. 5, the corrupted pixel is on l_m. Hence, the recovered pixel value can be simply obtained by

$$L_1 = \frac{\omega_1 X_m^* + \omega_2 Y_m^*}{\omega_1 + \omega_2}, \qquad (6)$$

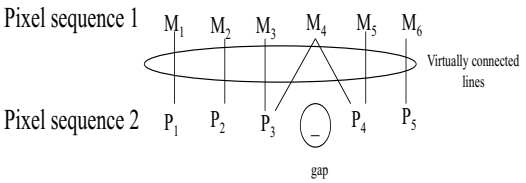

Fig. 4. The aligned pixels obtained from the example in Fig. 3.

where L_1 is the recovered pixel value, ω_1 and ω_2 are the distance between the corrupted pixel to Y_m^* and the corrupted pixel to X_m^*, respectively.

In the second case, the position of the corrupted pixel is in the closest location between two virtually connected lines. Therefore, the corrupted pixel is recovered by interpolating from the aligned pixels used to draw the underlying two virtually connected lines. For example, if the corrupted pixel lies between the connected lines l_m and l_n, where l_m is the virtual line connected between X_m^* and Y_m^* and l_n is the virtual line connected between X_n^* and Y_n^*. The recovered pixel value can be computed as

$$L_2 = \omega_1 X_m^* + \omega_2 X_n^* + \omega_3 Y_m^* + \omega_4 Y_n^*, \qquad (7)$$

where L_2 is the recovered pixel value, and $\omega_j, j = 1, \ldots, 4$ are the coefficients used for interpolation. The coefficients can defined as $\omega_1 = \frac{\omega}{d_1}$, $\omega_2 = \frac{\omega}{d_2}$, $\omega_3 = \frac{\omega}{d_3}$, and $\omega_4 = \frac{\omega}{d_4}$, where $\omega = \sum_{i=1}^{4} \omega_i$. d_1, d_2, d_3, and d_4 are distances between the location of the corrupted pixel to the positions of X_m^*, X_n^*, Y_m^*, and Y_n^*, respectively.

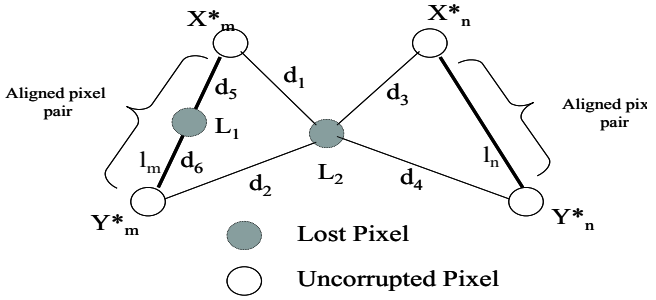

Fig. 5. The directional interpolation for the corrupted pixels from the surrounding aligned pixel pairs.

4. EXPERIMENTAL RESULTS

We conducted experiments to study the performance of the proposed error concealment algorithm in "Lena" image and "Mother and Daughter" video sequence. The proposed algorithm was applied to the luma component of image and video. We compared the concealment results obtained from the proposed algorithm and the bilinear interpolation approach [4]. Two image sequences used in algorithm are the pixel rows immediately above and below the corrupted region. The length of image sequence is equal to the width of the corrupted region.

For "Lena" image, we selected to randomly eliminate a 16x48 block of image (Fig. 6). For "Mother and Daughter" video sequence, the 16th GOB of the first intra frame was removed. The threshold used in (2) for the non-linear pattern alignment is set to 40. Fig. 6 and 7 showed the subjective and objective performance. We can see that ,with the non-linear alignment and directional interpolation, our proposed error concealment provides the improvement in recovering the corrupted region comparing to the bilinear interpolation. For objective measurement with PSNR, the proposed algorithm surpassed the bilinear interpolation equal to 1.18 and 0.8 dB for "Lena" and "Mother and daughter" image, respectively. In the subjective measurement, the proposed algorithm can preserve the texture and reduce the blurring effect, which is common in bilinear interpolation approach.

5. CONCLUSION

The new intra-frame error concealment algorithm based on the non-linear pattern alignment and the directional interpolation was presented. The non-linear pattern alignment approach was used to dynamically detect the possibly existed textures in the corrupted region. Then, the directional interpolation recovers pixels in the corrupted region based on the detected textures. From preliminary computer experiments, the proposed error concealment provided much better subjective and objective performance comparing to simple bilinear interpolation.

(A) PSNR = 41.9170 dB (B) PSNR = 43.0986 dB

Fig. 6. Subjective and objective quality of the concealed region of "Lean" image with A) The bilinear interpolation error concealment B) The proposed error concealment method.

(C) PSNR = 34.6 dB (D) PSNR = 35.4 dB

Fig. 7. Subjective and objective quality of the concealed region of "Mother and Daughter" image with C) The bilinear interpolation error concealment D) The proposed error concealment method.

6. REFERENCES

[1] S.B. Needleman and C.D. Wunsch, "A general method applicable to the search for similarities in the amino acid sequences of two proteins", *Journal of Molecular Biology*, vol. 48, pp. 443-453, 1970.

[2] J.-W. Suh and Y.-S. Ho, "Error concealment based on directional interpolation," *IEEE Trans. Consumer Electron.*, vol.43,pp.295-302,Aug. 1997.

[3] H. Sun and W. Kwok, "Concealment of damaged block transform coded images using projection onto convex set," *IEEE Trans. Image Processing,* vol. 4, pp. 470-477, Apr. 1995.

[4] M. Sun and A. Reibman, *Compressed Video over Networks*, Marcel Dekker Inc., 2001.

[5] W. Zeng and B. Liu, "Geometric-structure-based directional filtering for error concealment in image/video transmission," in *Proc. SPIE Conf. Wireless Data Transmission, Photonics East*, vol. 2601, Oct. 1995, pp. 145-156.