A RECURSIVE SHAPE ERROR CONCEALMENT ALGORITHM

Xiaohuan Li, A. K. Katsaggelos and *Guido M. Schuster (*)*

Image and Video Processing Laboratory, Department of Electrical and Computer Engineering, Northwestern University, Evanston, IL 60208
(*) Abteilung Elektrotechnik, Hochschule für Technik Rapperswil, Switzerland

ABSTRACT

The encoding of shape information is a distinguishing feature of MPEG-4. In error prone communication networks, it is important and efficient to conceal shape errors spatially, so as to avoid propagation of errors in the video frames. The proposed system first defines the missing area's four rectangular neighbors. It then detects in the neighbors the lines that intersect the borders and redefines them in a geometric coordinate system. These lines are paired under some optimization objective so that they connect smoothly in the missing area. Each pair is extrapolated and corrected recursively until a close curve is formed. Test results on various sequences under various error rates are presented and compared with other shape concealment methods.

1. INTRODUCTION

In addition to MPEG-1 and MPEG-2 standards' functionality of coding conventional images and video sequences, MPEG-4 enables a content-based representation of the audio and video data. In particular, it codes and decodes several multimedia objects separately, allowing greater flexibility and interactivity for standardized video and multimedia applications.

To synthesize different video objects (VO's), shape information is needed. A video object plane (VOP) is a temporal instance of a VO, defined by its texture and shape. Following an object-oriented approach, MPEG-4 visual transmits texture, motion, and shape information of one VO within one bit stream. The bit streams of several VO's and the corresponding composition information are multiplexed such that the decoder receives all the information to decode the VO's and arrange them into a video scene.

The shape of an object in MPEG-4 is defined by means of an alpha-plane, which defines the 'transparency' of an object and is not necessarily uniform. A binary alpha map defines whether or not a pixel belongs to an object (see Fig. 1 for an example). It can be 'on' or 'off'. A gray scale map offers the possibility to define the exact transparency of each pixel.

Inter-shape coding relies on a set of motion vectors (MV's), separate from texture's motion vectors. Within an error prone environment, motion compensation leads to increased sensitivity to channel noise and packet loss. The MPEG-4 standard has three built-in error-resilient techniques for shape information. 1) When the encoder enables the error-resilience mode, any pixel location that is outside the current video packet is transparent, so that error propagation across packets is prohibited. 2) Data partitioning separates the MB header, binary shape information, and MV's from the texture information. A resynchronization marker defines the boundary of the two partitions. The standard also specifies that if an error occurs in the texture part of a video packet, only the texture information is lost; shape and motion information from the other partition can be utilized to conceal texture at the same location. On the other hand, if an error occurs in the shape/motion part, the whole packet (shape, motion and texture) is lost. The importance of protecting/recovering shape information is apparent. 3) Insertion of a video packet header appears periodically in the bit stream and serves as a resynchronization marker, denoting the start of a MB. Moreover, the header contains information for decoding the received MB's, even if previous MB's are lost in the transmission.

Various approaches to conceal shape information have been proposed. However, most of them are based on a motion compensation scheme. These concealment methods are limited by the intrinsic setbacks of motion compensation, and perform poorly when objects appear/disappear, rotate or are distorted. Spatial concealment methods by nature bypass these problems. A MAP estimator with an MRF designed for binary shape information of neighboring blocks was proposed in [4]. It uses the spatial redundancy in transmitted shape information on a statistical base, and has not fully tapped into correlation and dependency between shapes of the neighboring areas.

2. PROPOSED METHOD

In this section we describe the recursive algorithm we have developed for shape error concealment. It is assumed that the alpha plane is in the binary mode and the missing area is of dimension 16x16 pixels, corresponding to one MB. The basic idea behind the proposed algorithm is first to find lines which extend into the to be concealed MB, second to best match up each line with another one and third to create curves in the to-be- concealed MB using these matching pair of lines. The line matching and curve creating steps are both conducted such that a smoothest contour is generated for the missing alpha block.

Step I: Context Detection

The off-line concealment system first locates the missing MB, as detected from the received video packets. The four rectangular neighboring MB's are read and recorded as *north*, *south*, *west* and *east* (as shown in Fig. 2). In order to take the diagonal neighbors into consideration, each block extends 4 pixels on the two sides (as shown in Fig. 2) and therefore becomes 24x16. A coordinate system is set for *south* as shown in Fig. 3. *North, west* and *east* are rotated to the position of *south* so as to use the same set of coordinates and concealment functions.

Next, lines that go over the x-axis (i.e. the border between a neighboring block and the missing block) are detected for each of the four neighbors: First, the 16x16 matrix is scanned row by row, and boundary points (0->1 or 1->0) points in each row are located. Second, line fitting is used on these points to generate a number of lines equal to the number of boundary points at the border between a neighboring block and the missing block, described by two parameters: *k* (slope) and *x0* (crossing position on x-axis). An example of this is shown in Fig. 3.

In step I, we extract parameterized geometric information of context shape out of the given binary data.

Step II: Pairing of Lines

Since the contour of a VO is a closed curve, each line that goes into a block must come out of it, i.e., $L_{all} = L_n + L_s + L_w + L_e$ (L_n is the number of lines crossing the north border, etc.) must be an even number. The L_{all} lines are divided into $L_{all}/2$ pairs so that each pair of lines is going to join up in the MB to be concealed. Let Φ be an ordered (counterclockwise) table of all boundary lines, i.e.:

$$\Phi = \{\underbrace{l_{s1}, l_{s2}}_{pair_1}, \ldots l_{sLs}, l_{e1}, l_{e2}, \ldots l_{eLe}, \underset{\ldots\ldots}{l_{n1}}, l_{n2}, \ldots l_{nLn}, l_{w1}, l_{w2}, \ldots, \underbrace{l_{w,Lw-1} l_{wLw}}_{pair_{Lall/2}}\}$$

where l_{s1} and l_{sLs} are the westmost and eastmost lines respectively from the south border. The slope of each line is then modified with its match as follows (see Fig. 4):

(1). Let $l_1 = \{k_1, x0_1\}$, $l_2 = \{k_2, x0_2, direction\}$ be a pair, where *direction* is the relative direction from l_2 to l_1, if they are not from the same border.

(2). Let θ_1 be the angle between l_1 and the corresponding border (x_1-axis), $\theta_1 \in (0, \pi)$.

(3). Let θ_2 be the angle between l_{1-2} and the x_1-axis, where l_{1-2} is the line determined by p_1 and p_2, p_1 being the crossing of l_1 and the x_1-axis and p_2 the crossing of l_2 and the x_2-axis.

(4). Assume that l_1 is from the south and l_2 from the east. As shown in Fig. 5, let d_1, d_2 be the distances from p_1 to the west border and diagonal, respectively, d_3, d_4 the distance from p_2 to the south border and diagonal, respectively, and d_5 the distance between p_1 and p_2.

As will become apparent later on the minimum of these distances $d = \min\{d_1, d_2, d_3, d_4, d_5\}$ determines the number of steps taken by the proposed algorithm. Then the factor $w=1/(d-1)$ is used to weigh the impact of l_2 on l_1; that is, smaller *d* is, the greater the impact of l_2 on l_1 (larger *w*). The choice of *w* accelerates joining of the two lines as their ends come closer, and guarantees convergence of the recursion, which will be discussed in part 3.

(5). Use the corrected slope $k_1' = \tan(\frac{\theta_1 + w\ \theta_2}{1+w})$, to obtain the modified amount $\Delta k_1 = k_1' - k_1$. Calculate the sum of squares of the modifications for all lines in Φ, denoted by $\Delta_1 = \sum_{l_i \in \Phi} (\Delta k_i)^2$.

Reorder Φ and list the lines clockwise, that is,

$$\Phi = \{\underbrace{l_{s1}, l_{wLw}}_{pair_1}, l_{w,Lw-1} \ldots, l_{w1}, l_{nLn}, \ldots, l_{n1}, \underset{\ldots\ldots}{l_{eLe}}, \ldots, l_{e1}, l_{sLs}, \ldots, \underbrace{l_{s3}, l_{s2}}_{pair_{Lall/2}}\}$$

Repeat the slope modification process ***(1)*** to ***(5)*** under the new pairing scheme. Calculate the sum of square of the new modifications for all lines, denoted by Δ_2.

By definition of Δ, we can see that the smaller Δ is, the less the slopes of the lines change in the recursion and the smoother the estimated contours are going to be. Namely, Δ measures the overall smoothness of the concealed shape. Therefore we adopt the pairing scheme under which Δ_i (*i*=1,2) is the smallest.

In step II, the context lines are paired up this way so that a smooth curve is mostly likely be generated later.

Step III: Recursive Concealment of a Pair of Lines

Let $l_1 = \{k_1, x0_1\}$, $l_2 = \{k_2, x0_2, direction\}$ be a pair under the adopted scheme from step II.

(1). Correct k_1 and k_2 with each other as described in step II. This leads to k'_1 and, k'_2.

(2). Predict the next point (x, 1) of l_1 within the to-be-concealed MB with the line equation $1 = k_1 x + b$, where $b = -k_1 \cdot x0_1$ is obtained by from the point $(x0_1, 0)$. The predicted point turns out to be $p_1 = (\frac{1 + k_1 \cdot x0_1}{k_1}, 1)$, the first point of the reconstructed curve. Predict the next point p_2 for l_2 likewise. This is the second point of the reconstructed curve.

(3). Find d the distance between the renewed points p_1 and p_2; if d is less than a threshold (say, 1), then quit the recursive process; otherwise increase the recursion counter by 1 and go to ***(4)***.

(4). Using the new points to calculate the new lines and new tangent slope k''_1 and k''_2. Proceed to ***(1)***. See Fig. 6 for the flow graph of the recursive concealment process.

Step IV: Fill in the Missing Area with Binary Colors

Convert the concealed lines coordinates in the missing block (an area of 16x16) into a 16x16 binary matrix. Binary color information has been passed on with the line information throughout the previous steps, now it can be used to recover the missing shape together with the line positions.

3. CONVERGENCE OF THE RECURSIVE ALGORITHM

As the recursive process proceeds, the two updated points p_1 and p_2 move closer to each other. As mentioned in the previous section the number of step of the algorithm equals $d = \min\{d_1, d_2, d_3, d_4, d_5\}$.

As the concealment proceeds, p_1 moves north. If it does not move toward connecting with p_2, it will exit through either the west border or the diagonal. Likewise, p_2 either joins p_1, or exits through the north border or the diagonal. At the *(d-1)* step, the weight *w* becomes infinite and $k'_1 = \tan(\theta_2)$. If the updated p_1 and p_2 have not met by this step, in the next step (last step) when l_1 is predicted, it is forced to the position of l_{1-2}, which links p_1 and p_2.

4. RESULTS

The performance of the concealment algorithm is measured by the ratio of the number of correctly concealed pixels over the number of missing pixels. Table 1 shows the performance of the proposed concealment system ("Line Method"), as well as, that of two existing ones ("MAP Method"[4] and "Median Method"). 30 realizations have been carried out for each error rate. See Fig. 7 for a comparison of visual concealed results.

5. CONCLUSIONS

The proposed shape concealment method summarizes shape information of a rectangular area in terms of boundary lines. It explores the known shape information on a relatively high level, and thus makes better use of the available information. The concealment system's performance achieves a 19.33% and 3.64% improvement over the traditional median method and the recent MAP method [4], respectively. Moreover, the line concealment method consumes far less computation resource than the MAP method.

6. REFERENCES

[1] Text for ISO/IEC FDIS 14496-2 Visual, ISO/IEC JTC1/ SC29/ WG11 N2502, Nov. 1998.

[2] J. Ostermann, et al., Coding of Arbitrarily Shaped Video Objects in MPEG-4, ICIP 1997

[3] J. Ostermann, Efficient Encoding of Binary Shapes Using MPEG-4, ICIP 1998

[4] Y. Wang, et al., Error Resilient Video Coding Techniques—Real-Time Video Communication Over Unreliable Networks, IEEE Signal Processing Magazine, Jul. 2000

[5] S. Shirani, et al., A Concealment Method for Shape Information in MPEG-4 Coded Video Sequences, IEEE Trans. on Multimedia, Sept. 2000

[6] V. Chen, et al., α -Channel Compression in Video Coding, ICIP 1998

[7] A. Puri, et al., *Multimedia Systems, Standards, and Networks*, Marcel Dekker, Inc, 1999

[8] P-J. Lee, et al. Robust Error Concealment Algorithm for MPEG-4 with the Aid of Fuzzy Theory, ICCE 2001

Error Rate	Line Method	MAP Method	Median Method
2.00%	95.50%	92.00%	40.00%
7.57%	95.77%	92.60%	92.47%
15.87%	95.59%	92.56%	88.78%
23.57%	98.73%	98.21%	97.49%
39.04%	96.06%	91.47%	89.86%
Average	95.57%	92.21%	80.09%

Table 1. Shape similarity measure for Bream frame0

Fig. 1. Alpha plane of frame0 of bream, VOP boundary is shown in white.

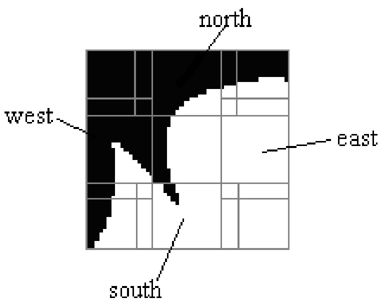

Fig. 2. North, east, south, west neighboring MB's used for the concealment of the MB at the center.

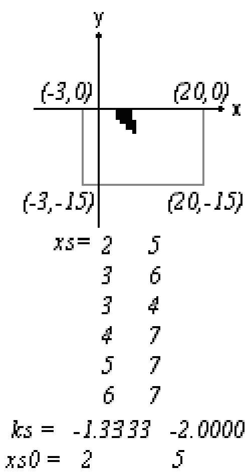

Fig. 3. . Coordinate system for *south* and lines detected from the southern border.

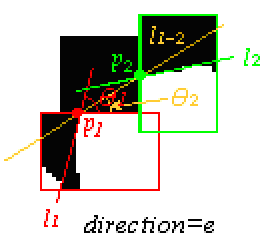

Fig. 4. Line correction between l_1 and l_2.

Fig. 7. (See right) Upper left: original of bream frame0; upper right: corrupted with 30% packet loss; lower left: concealed VOP by the proposed method; lower right: concealed VOP by the MAP method of [4].

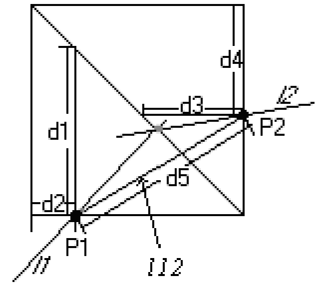

Fig. 5. Definition of d_1, d_2, d_3, d_4, d_5

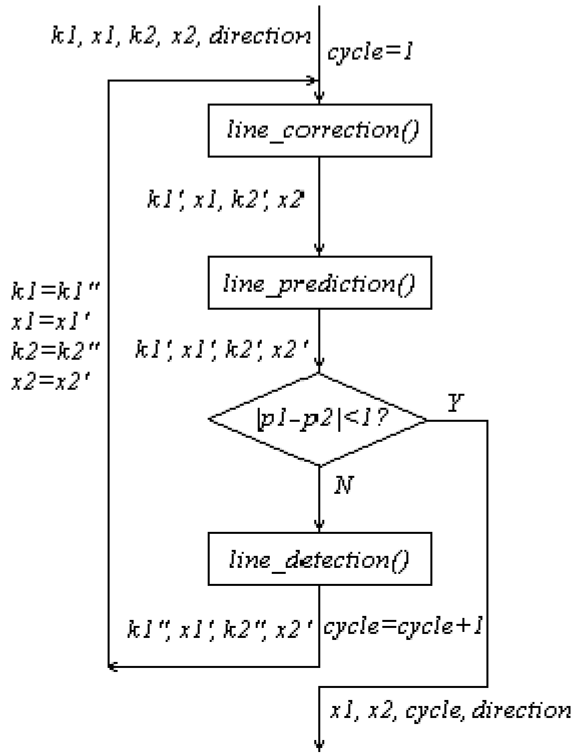

Fig. 6. Flow graph of the recursive concealment algorithm.