

ERROR CONCEALMENT FOR SHAPE CODING

Paul Salama and Cheng Huang

Visual Communications Lab
Department of Electrical and Computer Engineering
Indiana Univ. – Purdue Univ., Indianapolis

ABSTRACT

In ATM networks cell loss or channel errors can cause data to be dropped in the channel. When digital images/video are transmitted over these networks one must be able to reconstruct the missing data so that the impact of the errors is minimized. In this paper we present an error concealment technique for object-coded sequences using the MPEG-4 video coding standard. This method, which is based on using global motion estimation for boundary recovery, consists of boundary extraction, boundary patching using global motion compensation, and object filling to reconstruct the damaged video object.

1. INTRODUCTION

The MPEG-4 standard permits an object based audio-visual representation of data. Each video sequence may have several video objects. A video object at a given time instant is called a video object plane (VOP), which is the counterpart of a frame in the MPEG-1 and MPEG-2 standards [1, 2, 3]. VOPs can have arbitrary shapes and can be encoded independently of each other, or relative to each other by using motion compensation [1, 2].

The shape of a VOP is represented using a binary alpha plane [1, 2]. A binary alpha plane is binary image that represents the shape of the VOP. Pixels having value '1' represent the VOP and pixels with value '0' represent the background. The binary alpha plane is constructed in such a way that its dimensions are the smallest integer multiples of 16, and is divided into 16×16 blocks called Binary Alpha Blocks (BABs) [1]. Each BAB is encoded using a shape encoder that uses a block based context based arithmetic encoding (CAE) scheme combined only with motion compensation in the case of inter block BAB coding [1].

Unfortunately, compressed video objects are very vulnerable to channel errors and hence error concealment techniques are needed to circumvent the effect of data corruption or loss. In this paper we describe an error concealment technique for objects coded using the MPEG-4 video coding standard. The method, which is based on using global motion estimation for boundary recovery, consists of boundary extraction, boundary patching using

global motion compensation, and object filling to reconstruct the damaged video object.

2. ERROR CONCEALMENT FOR SHAPE IN MPEG-4 VIDEO

2.1 Previous work in shape error concealment

One of the techniques developed for shape error concealment is based on the modeling of shape via a binary Markov Random Field (MRF), and finding the maximum *a posteriori* (MAP) estimate of the missing boundary points [4, 5]. According to the algorithm, each missing pixel is estimated as a weighted median of the pixels in its clique [4, 5], where the weight corresponding to the difference between the pixel to be estimated and another pixel in its clique is selected based on the likelihood of an edge passing through in the direction of the subject pair of pixels. The rationale behind this selection is to weigh more the difference between the current pixel and a pixel in its clique in a direction along which pixels have a tendency to be the same. Although this method performed well, yet there exist some problems. First, the method makes use of only spatial redundancy to recover the missing pixels. Second, the method tries to recover the image data based on block recovery, not on boundary recovery, which may not always yield acceptable results. Third, the method can be computationally intensive.

2.2 Error concealment of shape using global motion compensation

In this section, a method for the error concealment of shape based on global motion compensation is proposed. It has been observed [6] that video sequences compressed using object-based compression are more error resilient if intra shape coding of the binary alpha blocks (BAB) in the video sequence is frequently used. This can effectively prevent error propagation, with an acceptable loss in compression efficiency. It has also been observed that the use of spatial information alone is not as effective in performing error concealment as the use of temporal data [6]. To prevent error propagation and to provide resynchronization, all BABs are intra coded. This, however, requires that extra temporal data be inserted into the data stream.

A problem that arises is how much extra data is needed for concealing errors without compromising compression efficiency. Fortunately, since shape coding deals with boundaries, we need not trace the motion of all the macroblocks belonging to the object. Only the motion of the object boundary is required. Hence global motion estimation is used to trace the motion of the object. The corresponding motion vectors are utilized at the decoder to perform error concealment.

When using global motion estimation to perform shape error concealment, it is only the boundary of a video object plane that is of importance. In addition, only the following aspects of camera motion are considered [6, 7]: Change of the camera focal length (zoom or scale), rotation around the camera axis, and translation in the plane normal to the camera.

2.3 Global Motion Parameters

To represent global motion, four parameters are used. Two parameters are used for the centroid of the object, the third parameter is for scale or zoom, and the fourth is for the rotation angle [6, 7]. These are obtained as in [7] and inserted into the MPEG-4 bit stream using the USER_DATA field. The resulting increase in stream size was observed to be 5%.

The steps involved in recovering object shape are: (1) Extracting the boundary of the current VOP. (2) Extracting the boundary of the reference VOP, if the boundary of the current VOP is not intact due to data loss, and using it to reconstruct the boundary of the current VOP. (3) Filling in the closed boundary of the current VOP with opaque pixels to fully reconstruct the current VOP.

2.4 Boundary extraction

When a VOP is detected as being damaged, the first step is to extract its boundary. There are two methods that can be used to determine whether a pixel is a boundary pixel or not [7]. The first method uses a 4-neighborhood, which results in an 8-connected boundary. The second method uses an 8-neighborhood, which results in a 4-connected boundary, to make decision [7]. Due to the ease of traversing a 4-connected boundary, the 8-neighborhoods are chosen to perform boundary extraction. This method however does not always perform well. Figure 1 shows part of a boundary extracted from a corrupted VOP by exploiting the 4-neighborhoods. It can be seen that the boundary is not simple enough to traverse. When traversing to the valley of this curve, it is not clear how to proceed because some boundary pixels at the valley of the curve have more than two neighboring boundary pixels in the sense of 4-connectedness. This problem is overcome by applying median filtering to the corrupted VOP before extracting its boundary. A 5-tap 1-D median filter is used. The process is as follows: (1) Apply the 1-D median filter

to the rows of the image. (2) Apply the 1-D median filter vertically to the columns of the image resulting from step (1). (3) Compare the image resulting from step (2) and the image prior to step (1): if there is no difference, terminate, and the resulting image is used for boundary extraction; otherwise, go to step (1) and repeat. Experimental results have shown that this method makes the extracted boundary simple enough to be traversed for most of the images, and on average 2 to 3 iterations were needed for a QCIF image.

2.5 Boundary reconstruction by global motion compensation

After extracting the boundary of the corrupted VOP, it has to be decided whether boundary reconstruction is needed or not. If the corrupted binary alpha blocks do not cross the boundary, the extracted boundary is then intact and does not require reconstruction. The criterion for deciding whether the extracted boundary is closed or not is the number of endpoints. If the number of endpoints is zero, then the boundary is considered closed. The process for reconstructing the damaged boundary via global motion compensation can then be divided into following steps:

- (1) Partitioning the endpoints in the current VOP into pairs such that each pair of endpoints belongs to one separate part of the boundary, see Figure 3. Note that the total number of endpoints is always even.
- (2) Extracting the closed boundary of the reference VOP, and mapping the endpoints in the current VOP to the boundary pixels in the reference VOP [6], using the global motion parameters, see Figure 4.



Figure 1: Part of the extracted boundary.



Figure 2: Part of the boundary extracted after iterative median filtering of the corrupted VOP.

- (3) Traversing the boundary of the reference VOP starting from endpoint 1 along the direction of $1 \rightarrow 1'$, and recording the order of the endpoints traversed. For example, the order of the endpoints traversed in Figure 4 is $(1 \rightarrow 1' \rightarrow 3' \rightarrow 3 \rightarrow 2 \rightarrow 2')$.



Figure 3: Partitioning endpoints in the damaged boundary of the current VOP.

- (4) Traversing the boundary of the reference VOP according to the order just derived from step (3). For each curve whose endpoints belong to different pair, for example $1' \rightarrow 3'$, each pixel in this curve is mapped to the boundary image of the current VOP [6]. In the boundary image of the current VOP, each mapped boundary pixel is connected to the already existing boundary using bi-linear interpolation, which results in a closed and patched boundary.



Figure 4: Mapped endpoints in the boundary of the reference VOP.

2.6 VOP recovery by filling in the closed boundary

After the damaged boundary is reconstructed, the final step is to fill in the closed boundary with opaque pixels that make up the VOP. The algorithm employed for filling in the closed boundary is the lexicographic seed-filling algorithm. This algorithm needs a stack of stored seeds (pixels), and an initial seed to initialize the filling process. The algorithm is as follows: (1) Initially pick a pixel inside the boundary as the initial seed and push it into the empty stack. In our method, the centroid is used as the seed. Experimental results have shown that in most cases, the centroid is inside the VOP. (2) Decide whether the stack is empty. If the stack is empty, the filling process is over and terminate; otherwise, go to step (3). (3) Pop a seed from the stack, and set it to be opaque. Then traverse to the left and right along the line on which the seed lies, until the boundaries of the VOP are met. (4) Record the boundary pixels met in the scan as x_l and x_r . This process is known as "line scan". Each pixel traversed during the line scan is set to opaque. (5) Scan the line right above the current line and find the left and right boundary points, x_{l1} and x_{r1} respectively. Find also all the pixels between x_{l1} and x_{r1} that belong to the object. Denote these by $[x_{l1}, x_{r1}]$. For each $[x_{l1}, x_{r1}]$, decide whether the range of the pixels

overlaps with the set of pixels $[x_l, x_r]$ on the current line. If so, push the right most pixel in the range $[x_{l1}, x_{r1}]$ onto the stack. (6) Apply the same process in step (4) to the line right below the current line. (7) Goto step (2).

2.7 Computational Complexity

To analyze the complexity of our algorithm we assume that the dimension of the VOP is $n \times n$. The computation needed in our method involves: (1) boundary extraction which requires $o(n^2)$ computation; (2) boundary patching which requires $o(n)$ computation; (3) filling in the boundary which requires $o(n^2)$ computation. Thus, the computation complexity for our method is $o(n^2)$.

The statistical method based on the Markov Random Field (MRF) model is iterative. The computational load of each iteration involves: (1) finding the boundary pixels in the neighboring blocks; (2) calculating the slope of the best line-fit for each of the boundary pixels; (3) estimating missing pixels which involves median filtering. It can be shown that the computation complexity for the statistical method is also $o(n^2)$. However, the global motion compensation method may be more efficient due to the following reasons: (1) According to [4, 5], the number of iterations for the statistical method is around 5 or 6, which is greater than that for our method. (2) The iteration in the statistical method is conducted at every step while the iteration in our method is only for the median filtering. (3) The error pattern influences the computation required in patching, but does not affect the computation for boundary extraction and VOP filling, which account for most of the computation. Thus, the computational load in our method is quite stable. On the other hand, the statistical method is block-based, and the type of error pattern may result in a large variation in computation.

3. EXPERIMENTAL RESULTS

To test our algorithm we compressed a QCIF sized video sequence to a data rate of 122.4 kbps using video packet sizes of 384 bytes. The video packets were packed into ATM cells that were transmitted, and subjected to a random loss of 3%. The effect of global motion compensation method for error concealment on different error patterns is shown in Figure 5 and Figure 6. In Figure 5 we show the performance of our technique when 2 cells were lost at a time, and in Figure 6 we demonstrate the performance of our method when 4 cells were lost at a time. As observed the experimental results reveal that using global motion to perform error concealment can produce good results for a large range of error patterns. In all cases, it is assumed that the reference VOP is undamaged or has been fixed. If the reference VOP is corrupted, error concealment is initially performed on the reference VOP first. If the reference VOP cannot be corrected then the current VOP is skipped. If a sequence of VOPs is damaged,

shape recovery commences at the first undamaged VOP and proceeds until the last damaged VOP is recovered.

4. CONCLUSION

In this paper we have described a shape based error concealment technique that utilizes temporal correlation between adjacent VOPs and global motion data to extract the boundary of a damaged VOP, map it to the boundary of a reference VOP, and reconstruct the damaged boundary. After the boundary of an object has been recovered, the object is filled in.

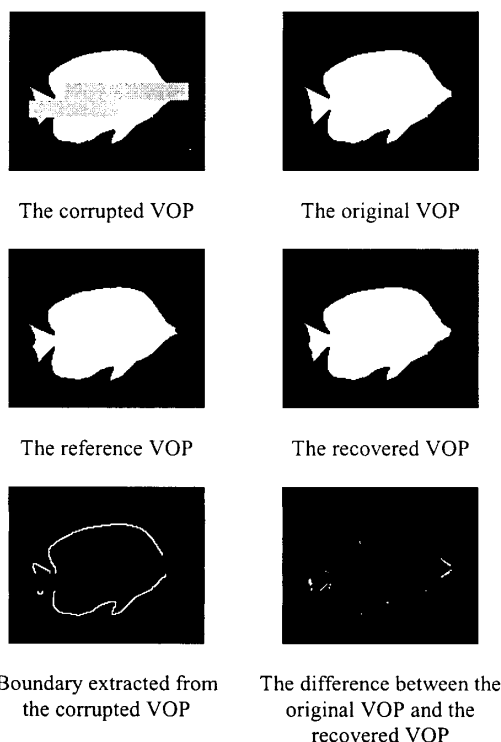


Figure 5: Error concealment for a cell loss rate of 3%, a data rate of 122.4 kbps, and video packet size of 384 bytes when 2 cells were dropped consecutively.

5. REFERENCES

[1] MPEG Video Group, "MPEG-4 video verification model 18.0," Doc. ISO/MPEG N3908, Pisa, January 2001.

[2] T. Ebrahimi, and C. Horne, "MPEG-4 natural video coding – An overview," *Signal Processing: Image Communication*, vol. 15, pp. 365-385, 2000.

[3] J. Ostermann, E. Jang, J. Shin, and T. Chen, "Coding of arbitrarily shaped video objects in MPEG-4," *Proceedings of IEEE International Conference on Image Processing*, Santa Barbara, October 1997.

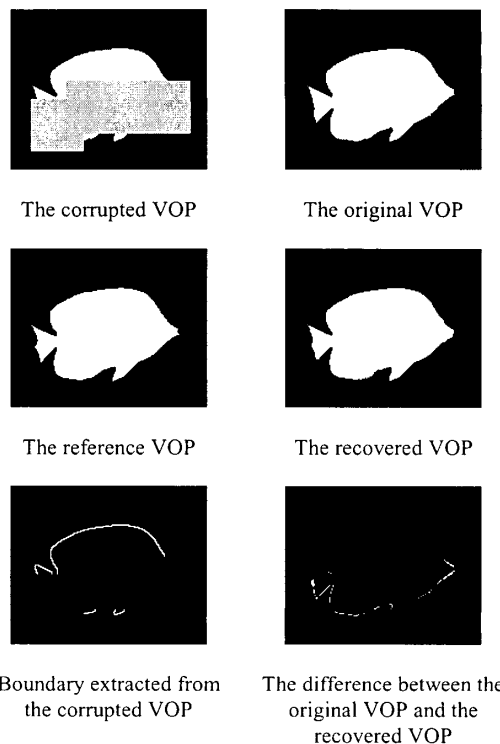


Figure 6: Error concealment when the cell loss rate is 3%, data rate is 122.4 kbps, and video packet size is 384 bytes when 4 cells were dropped consecutively.

[4] S. Shirani, B. Erol and F. Kossentini, "Error concealment for MPEG-4 video communication in an error prone environment", *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 2107-2110, 2000.

[5] S. Shirani, B. Erol, and F. Kossentini, "Concealment method for shape information in MPEG-4 coded video sequences," *IEEE Transactions on Multimedia*, vol. 2, no. 3, September 2000, p 185-190.

[6] C. Huang "Error Concealment for shape in MPEG-4 natural video", *Master's Thesis*, IUPUI, 2001.

[7] Nello Zuech, Richard K. Miller, *Machine Vision*, Van Nostrand Reinhold, 1989.

Address all correspondences to Paul Salama, salama@ecn.purdue.edu, <http://www.engr.iupui.edu/~salama/>, or +1 317 278 1682.