MOTION-BASED SHAPE ERROR CONCEALMENT FOR OBJECT-BASED VIDEO

Luis Ducla Soares, Fernando Pereira
Instituto Superior Técnico / Instituto de Telecomunicações

ABSTRACT

In this paper, an original motion-based shape error concealment technique, especially useful for object-based video applications in error-prone environments such as mobile networks, is proposed. It is assumed that the shape of the corrupted object at hand is in the form of a binary alpha plane and some of the shape data is missing due to channel errors. To conceal the corrupted shape, the decoder starts by assuming that the alpha plane changes in consecutive time instants can be described by a global motion model. This way, based on locally estimated global motion parameters, the decoder tries to conceal the corrupted alpha plane by global motion compensating the shape data from the previous time instant. Then, since not all alpha plane changes can be perfectly described by global motion, an additional local motion refinement is applied to deal with areas of the object that have significant motion.

1. INTRODUCTION

The appearance of the MPEG-4 object-based audiovisual coding standard [1] opened up the way for new video services, where scenes are understood as a composition of objects. However, in order to make these object-based services available in error-prone environments, such as mobile networks or the Internet, with an acceptable quality, appropriate error concealment techniques dealing with both shape and texture data are necessary.

While (frame-based) texture concealment schemes are abundant in the literature and can with more or less adjustments be adapted to work for object-based video, shape concealment schemes are just starting to appear. In particular, the method proposed in [2] is based on the idea that the contour changes (i.e., the boundary of the shape data) for a given video object, in consecutive time instants, can be described by a global motion model. Based on this assumption, the global motion parameters are estimated at the encoder and sent along with the encoded bitstream to the decoder. This way, when errors occur and cause the contour to be broken in several places, the decoder can easily (global) motion compensate the contour from the previous time instant using the information sent by the encoder and restore the corrupted contour. To recover the shape, the decoder has simply to fill in the concealed contour.

The first drawback of the technique described above is that the global motion parameters are computed at the encoder, when the sequence is being encoded. These parameters are transmitted to the decoder using a separate stream, which according to [2] may represent about a 5% bit rate increase. This is a serious limitation because the technique can only be used if both the encoder and decoder support it, which is very unlikely in such a competition-driven market with many manufacturers since this type of stream is not normatively specified. In addition, other issues must be considered, such as how to synchronize this new stream with the visual data stream and what should be done if this stream is also corrupted. The second drawback of the type of technique described above is that it considers only global motion compensation to do the concealment. Because of this, it is not able to adequately deal with shapes that have some local motion, which significantly limits its concealment performance for less constrained shapes.

2. PROPOSED MOTION-BASED SHAPE ERROR CONCEALMENT ALGORITHM

In this paper, the idea of using global motion to conceal shape errors at the decoder is extended while overcoming the two major drawbacks mentioned above. In the proposed technique, all error concealment related processing is performed at the decoder, which no longer has to rely on additional (non-normative) information sent by the encoder. This allows the proposed technique to be used even if the encoder knows nothing about the concealment techniques implemented at the decoder, which is usually the case when terminals from different manufacturers are used, even if the same standard coding format is used. This way, the additional bit rate that was used in [2] for the extra stream carrying the global motion parameters can be used for something else, like increasing the shape intra refreshment rate or improving the texture quality. In addition to this, and in order to deal with shapes that have some local motion, the proposed concealment technique also includes a local motion refinement step.

In this paper, it is assumed that the alpha planes have been encoded with some kind of block-based technique before being delivered, such as (but not necessarily) the solutions specified in the MPEG-4 Visual standard [1]. It is also considered that bitstream errors will manifest themselves in the form of bursts of consecutive erroneous 16×16 blocks, which is the most common case, at least in video coding standards.

Since the global motion parameters are no longer sent by the encoder, they have to be locally computed at the decoder. This is only possible because the decoded video data at a given time instant is usually not completely corrupted, since errors typically manifest themselves as bursts of consecutive erroneous blocks. With the remaining correctly decoded shape and texture data, the decoder can extract the necessary global motion parameters.

After the global motion parameters have been determined, they can be used to motion compensate the alpha plane of the previous time instant. This way, the erroneous shape blocks in the corrupted alpha plane being concealed can be simply replaced by the co-located shape blocks in the motion compensated previous alpha plane. Assuming that the global motion model can accurately describe the shape motion, this concealment alone should be able to produce rather good results. However, in many cases, this does not happen, due to the existence of local motion in some areas of the shape. Therefore, to avoid significant shape artifacts when concealing erroneous blocks in areas with local motion, an additional local motion refinement scheme has been introduced in the concealment process. In this scheme, the available blocks surrounding an erroneous shape block are used to determine if any local motion exists; if so, a local motion vector is estimated and used to find a better replacement block from the previous alpha plane for the erroneous shape block.

The block diagram for the proposed temporal shape error concealment technique is presented in Figure 1, where each block corresponds to one of the three consecutive steps in the concealment process.

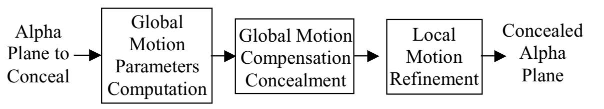

Alpha Plane to Conceal
Global Motion Parameters Computation
Global Motion Compensation Concealment
Local Motion Refinement
Concealed Alpha Plane

Figure 1 – Motion-based shape error concealment architecture

In order to understand better the shape concealment process, an illustrative example will be given before detailing each of the steps above in the following sections. In this example, the alpha plane in Figure 2 (a) has been corrupted, as illustrated in Figure 2 (b). Based on the estimated global motion parameters (using the information in Figure 2 (b)), the previous VOP is motion compensated and Figure 2 (c) is obtained. After that, the corrupted blocks are replaced with the corresponding ones in the global motion compensated previous VOP, which gives the concealed alpha plane shown in Figure 2 (d); this concealed alpha plane shows significant artifacts in the areas where significant local motion exists, notably in the racket zone. After the local motion refinement, the concealed alpha plane in Figure 2 (e) is finally obtained.

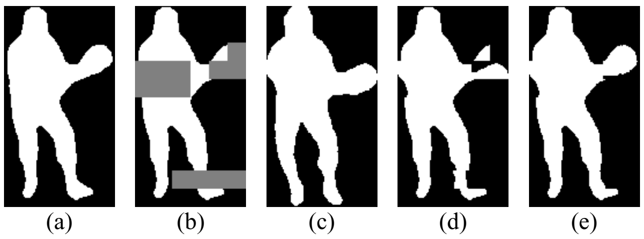


Figure 2 – Temporal shape concealment process for the Stefan video object: (a) Original uncorrupted alpha plane; (b) Corrupted alpha plane; (c) Global motion compensated previous alpha plane; (d) Concealed alpha plane without local motion refinement; (e) Concealed alpha plane with local motion refinement

2.1 Global motion parameters estimation

Before computing the global motion parameters, a global motion model has to be chosen from the ones available. The major difference between existing motion models is basically related to their complexity and, thus, their capacity to describe complicated motion trajectories. Since the motion of the shape data between consecutive time instants is typically quite simple, a simple model should be enough. However, it is always possible to replace the selected motion model with a more complex one, since the proposed technique does not directly depend on the type of motion model that is used, as long as one is used. This way, the affine four parameter model, which is certainly the simplest and most widely used global motion model, was selected. A description of how this model is derived can be found in [3], as well as the types of motion that are possible to describe with it, notably: i) change of the camera focal length (i.e., zoom or scale); ii) rotation around an axis normal to the camera axis (i.e., pan); and iii) rotation around the camera axis.

With this model, when two time instants and forward motion are considered, the transparency value of a shapel with coordinates (x',y') in the most recent time instant can be computed from the shapel with coordinates (x,y) in the previous time instant by the following expression:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} c_1 & c_2 \\ -c_2 & c_1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} c_3 \\ c_4 \end{bmatrix} \tag{1}$$

where c_1, c_2, c_3 and c_4 are the global motion parameters.

Since the alpha plane is completely uniform on either side of the video object contour (opaque inside the object and transparent outside), it does not have any internal motion and, therefore, its motion basically corresponds to the motion of the contour. This way, to determine the motion of the shape data, the first step is to extract what is left of the video object contour from the correctly decoded shape data.

The next step is to determine, for each point of the extracted contour, the corresponding point in the previous alpha plane; this is the same as determining a motion vector for each contour point. For this, a shape context around the considered contour point is considered; the used context is a block of 16×16 shapels and the search range is 32 shapels (16 to each side) in both directions, but these values can easily be changed. Here, the accuracy of the estimation is favored and, therefore, an exhaustive search pattern is used. However, this could be replaced with a faster, although sub-optimal, algorithm such as a three step hierarchical search [4]. This search for the corresponding point in the previous alpha plane can be quite a daunting task because, in many cases, several perfectly matching candidates can be found, leading to disastrous results in terms of global motion parameters. Therefore, to improve the global motion estimation, a 16×16 luminance context around the considered contour points will be used because the shape context alone gives too many inconsistencies, leading to rather inaccurate motion parameters.

After all the corresponding points have been found for the current and previous alpha planes, the global motion parameters are determined by finding linear least squares estimates of c_1, c_2, c_3 and c_4, as described in [3].

2.2 Global motion compensation concealment

After the global motion parameters are known, the motion compensation itself is quite straightforward. The objective of this module is to take the correctly decoded (or partly concealed) alpha plane from the previous time instant and global motion compensate it to the current time instant, so that it can be used to conceal the corrupted parts of the current alpha plane. To do this, all the decoder has to do is consider all the points with coordinates (x,y) in the previous alpha plane and compute their new coordinates (x',y') in the current time instant where the concealment is to be applied, by using Equation (1). To make the motion compensation more efficient, only the points with coordinates (x,y) that correspond to opaque shapels in the previous alpha plane have to be considered. Of course, motion compensation can be applied to both shape and texture data.

After the previous alpha plane has been motion compensated, the concealment itself, which is simply a cut and paste operation, can start. In the corrupted alpha plane, all the corrupted shape blocks are replaced with the corresponding alpha plane blocks from the motion compensated previous alpha plane. The same can be done for the texture data.

To better understand this procedure, Figure 3 should be considered. In Figure 3 (a), the corrupted alpha plane is shown. In Figure 3 (b), the previous global motion compensated alpha plane is shown, where the shaded areas correspond to the corrupted alpha blocks in the current corrupted alpha plane. By simply copying these shaded blocks to the corrupted alpha plane, Figure 3 (c) is obtained, where the concealed blocks are shaded.

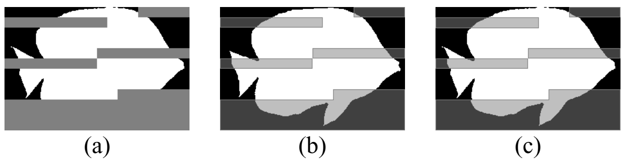


Figure 3 – Replacement of corrupted alpha plane blocks: (a) Corrupted alpha plane; (b) Previous global motion compensated alpha plane; (c) Concealed alpha plane

2.3 Local motion refinement

As illustrated in Figure 2, for alpha planes that have significant local motion, the global motion compensation concealment process described in Section 2.2 is not enough to produce good results. Therefore, to deal with the areas that have local motion, a refinement is necessary. However, before trying to refine the proposed concealment method, it is important to distinguish two types of concealed blocks: blocks that have at least one correctly decoded neighboring (shape) block and blocks that only have neighboring blocks that have been (shape) concealed themselves. Since the former have much more reliable neighboring shape data (because it has been correctly decoded), the refinement should start with those. The local motion concealment refinement is applied to the latter blocks only afterwards. Thus, the refinement procedure consists of two consecutive steps described in the next two sections.

2.3.1 Local motion refinement of shape blocks with correct neighbors

To refine the concealment of blocks that have at least one correctly decoded neighboring block, the shape blocks in the concealed alpha plane are scanned from top to bottom, left to right. For each concealed shape block that has at least one correctly decoded neighboring block, the decoder has to determine if it has been acceptably concealed with the global motion compensation alone. This is done by inspecting the correctly decoded neighboring shape blocks and comparing them with the shape blocks that would have been obtained if global motion compensation concealment had also been applied to them. If the number of different shapels does not exceed 90 for all the correctly decoded neighboring shape blocks and 30 for any individual block, the shape block at hand is considered to have been adequately concealed and no further processing is needed for it. Otherwise, local motion refinement will be applied. For this, a motion vector is determined for each one of the non-transparent correctly decoded shape blocks surrounding the shape block in question. This is done by applying a typical block-matching motion estimation algorithm to the current luminance plane and the previous luminance plane. Then, local motion compensated concealment is tried for the shape block at hand with the determined motion vectors (i.e., replacing the corrupted shape block with the shape block from the previous alpha plane indicated by the determined motion vector). In addition to the various individual motion vectors, the average motion vector is also tried. If some of the neighboring shape blocks were not correctly decoded but have already been (concealed and) local motion refined and, therefore, already have local motion vectors associated to them, they are also included in the computation of the average motion vector. From the tested motion vectors, the one that will be used to perform the final local concealment is the one that minimizes a shape border continuity metric (SCM) for the concealed shape block. This metric is defined as:

$$SCM = \sum_i |s_i - \hat{s}_i| \tag{2}$$

where s_i corresponds to the shapels in the four borders of the concealed shape block being refined and $\hat{s}_i$ corresponds to the shapels across the border from s_i. To compute the SCM, only the neighboring blocks that have either been correctly decoded or already undergone local refinement are taken into account.

2.3.2 Local motion refinement of shape blocks with no uncorrupted neighbors

To refine the concealment of the shape blocks that do not have any correctly decoded neighboring blocks, the shape blocks in the concealed alpha plane are scanned from top to bottom, left to right. For each concealed shape block that does not have any correctly decoded neighboring block, the decoder has to determine if any of the neighboring shape blocks have been locally refined. If not, nothing is done at this time for this shape block. If, on the other hand, some of the neighboring shape blocks have already been locally refined, their previously determined (i.e., in the previous section) motion vectors should be considered and used to perform local motion compensated concealment for the shape block at hand. As in Section 2.3.1, the average motion vector is also tried. To choose which motion vector will be used to perform the final concealment, the metric in Equation (2) will also be used. The same motion vector can also be used to refine the concealed texture data.

In order to guarantee that all the necessary shape blocks are refined, the procedure described above has to be performed again from bottom to top, right to left.

3. PERFORMANCE EVALUATION

In order to evaluate the proposed motion-based shape concealment technique, the *Akiyo*, *Bream* and *Stefan* video objects have been encoded according to the MPEG-4 Core Visual Object Type. In terms of MPEG-4 video error resilience tools, resynchronization markers and data partitioning with reversible variable length codes were used. Additionally, since it was important (for the estimation of global motion parameters) to avoid that the decoded texture quality degrade too much, a periodic intra refreshment scheme was used at the encoder. As for the shape data, it was intra coded at each time instant.

Here, instead of adding errors directly to the bitstreams, the decoder simply ignored the video packets randomly with a given packet loss rate (following a uniform distribution which produces bursts of corrupted shape blocks), thus allowing to evaluate the performance of the concealment technique independently of the decoder error detection capabilities. For each one of the studied loss rates, each video object has been decoded 50 times (i.e., corresponding to 50 different error patterns or runs), while applying the proposed motion-based shape error concealment technique to the corrupted alpha planes.

To evaluate the shape quality, the *Dn* metric used by MPEG is adopted, which is defined as the number of different shapels between the decoded and original alpha planes divided by the total number of opaque shapels in the original alpha plane. Additionally, since the proposed technique can also be applied to the texture of the video object, whose decoded texture quality is here very important for the estimation of the global motion parameters, numerical texture quality results will also be presented using the *PSNR* metric. However, since arbitrarily shaped video objects are used, the *PSNR* metric is only computed over the pixels that belong to both the decoded VOP being evaluated and the original VOP.

Due to space limitation, results are only shown here for the CIF version of the *Stefan* video object; the lowest acceptable frame rate for this sequence (i.e., 15 fps) was used because it corresponds to the most critical situation in terms of temporal error concealment. At this frame rate, an acceptable texture quality can be obtained by encoding the sequence at 128 kbps. As for the video packet size, this was chosen to be 1060 bits, which corresponds to eight video packets per VOP. With these parameters, an average error-free decoded texture quality of 30.46 dB is obtained. The obtained *Dn* for error-free conditions is obviously 0.00% because MPEG-4 shape coding is lossless.

In Table 1, Dn_{low} and Dn_{high} correspond, respectively, to the average *Dn* values associated with the best and the worst runs in terms of shape quality. As for Dn_{avg}, it corresponds to the mean of the average *Dn* values associated with the 50 different runs for each test case. In Table 2, $PSNR_{low}$, $PSNR_{avg}$ and $PSNR_{high}$ have equivalent definitions.

Table 1 – Dn values for the Stefan video object

Video packet loss rate	Dn_{low} [%]	Dn_{avg} [%]	Dn_{high} [%]
1%	0.09	0.32	0.57
5%	0.92	1.67	2.55
10%	2.25	3.42	5.09
20%	6.17	7.45	9.79

Table 2 – PSNR values for the Stefan video object

Video packet loss rate	$PSNR_{low}$ [dB]	$PSNR_{avg}$ [dB]	$PSNR_{high}$ [dB]
1%	27.77	28.99	29.89
5%	23.82	25.05	26.20
10%	21.35	22.50	23.66
20%	18.94	19.59	20.40

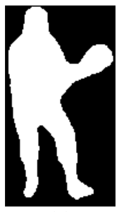
As expected, Table 1 shows that the *Dn* values increase gradually as the packet loss rate increases, going from values well below 1% to values above 5%. Notice that *Dn* values below 1% correspond to hardly noticeable artifacts, while for values above 5% the artifacts start to become quite visible. As for the *PSNR* values in Table 2, they gradually decay as the packet loss rate increases, which was also expected.

In order to illustrate the results in Table 1 and Table 2, the shape and texture of a given decoded VOP (VOP 3 in the original 300 VOP sequence) is used. In Figure 5, three different corrupted versions of the alpha plane in Figure 4 (a) and the texture in Figure 4 (b) are shown, in addition to the corresponding concealed alpha planes and textures. These three versions correspond to three of the 50 different error patterns for a video packet loss rate of 20% (the worst tested). As can be seen in Figure 5, although some artifacts are visible, the obtained results are visually quite acceptable taking into account the complexity of the shape and the amount of errors. The *Dn* values for the concealed alpha planes shown are: 11.50% for error pattern 1, 8.86% for error pattern 2 and 7.58% for error pattern 3. As for the corresponding *PSNR* values, they are 24.41 dB for pattern 1, 26.91 dB for pattern 2 and 21.91 dB for pattern 3. The uncorrupted texture has a *PSNR* value of 30.31 dB.

4. FINAL REMARKS

In this paper, a temporal motion-based technique was proposed to conceal shape errors in binary alpha planes or in the binary support of gray scale shapes for object-based video coding systems, such as those based on the MPEG-4 standard. Results have been presented showing the ability of this technique to recover lost shape data with rather small distortion, even for cases where local motion exists and the global motion model alone is not able to perfectly describe the shape motion.

Finally, it is important to emphasize the relevance of shape concealment techniques, not only to achieve an acceptable shape quality, but also because the decoded texture quality obtained is highly dependent on the quality of the shape data (i.e., the texture data can only be correctly decoded if the shape data is correct). Therefore, for object-based video applications to be actually deployed in error-prone environments, robust shape error concealment techniques will have to be available.


(a)

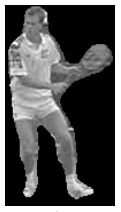
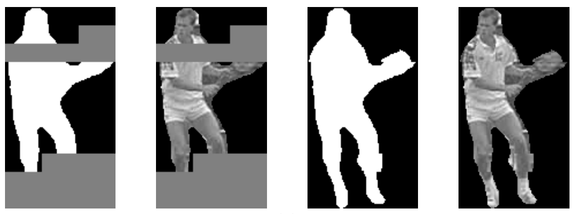
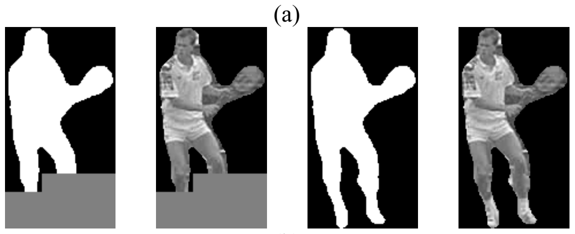

(b)

Figure 4 – Stefan video object: (a) alpha plane; (b) texture


(a)


(b)

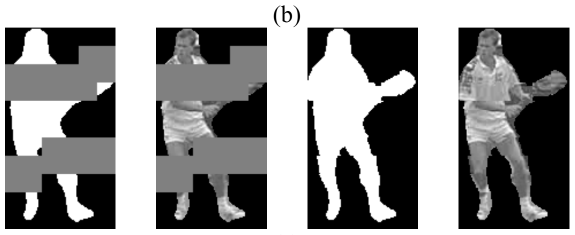

(c)

Figure 5 – Corrupted and respective concealed alpha planes and textures for the Stefan video object with a video packet loss rate of 20% for: (a) error pattern 1; (b) error pattern 2; (c) error pattern 3

5. ACKNOWLEDGMENT

The work presented was developed within VISNET, a European Network of Excellence (http://www.visnet-noe.org).

6. REFERENCES

[1] ISO/IEC 14496-2, "Information Technology – Coding of Audio-Visual Objects, Part 2: Visual," Dec. 1999.
[2] P. Salama, C. Huang, "Error Concealment for Shape Coding," *Proc. of ICIP 2002*, Rochester, NY, USA, Vol. 2, pp. 701-704, Sep. 2002.
[3] A. Zakhor, F. Lari, "Edge Based 3-D Camera Motion Estimation with Application to Video Coding," *IEEE Trans. on Image Proc.*, Vol. 2, No. 4, pp. 481-498, Oct. 1993.
[4] M. Bierling, R. Thoma, "Motion Compensating Field Interpolation Using a Hierarchical Structure Displacement Estimator," *Signal Processing*, Vol. 11, No. 4, pp. 387-404, Dec. 1988.