SPATIAL TEXTURE ERROR CONCEALMENT FOR OBJECT-BASED IMAGE AND VIDEO CODING

Luis Ducla Soares[1], Fernando Pereira[2]

[1]Instituto Superior de Ciências do Trabalho e da Empresa / Instituto de Telecomunicações
[2]Instituto Superior Técnico / Instituto de Telecomunicações
[1]lds@lx.it.pt, [2]fp@lx.it.pt

Abstract: *In this paper, an original texture error concealment technique to be used for object-based image and video applications in error-prone environments is proposed. This technique, which assumes that the texture is block-based, addresses the specific problems of texture concealment in object-based coding systems such as those based on the MPEG-4 standard. For that, the closest available blocks to each side are used to estimate the lost texture data in the corrupted area. Two different approaches are compared: one based on the linear interpolation of the available data and another based on the weighted median.*

Key words: *spatial concealment, texture concealment, object-based image and video coding.*

1. INTRODUCTION

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; this approach may have advantages in terms of coding efficiency as well as in terms of additional functionalities. However, 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.

In terms of shape data concealment, several techniques have already been proposed in the literature, which can be divided in two major categories, depending on the information that is used for the concealment process. On one hand, spatial error concealment techniques, such as those in [2][3], only use spatially adjacent shape information to perform the concealment. This makes these techniques especially useful when the shape changes greatly in consecutive time instants, such as when new objects appear or are uncovered. Additionally, they can also be used for still images. On the other hand, temporal error concealment techniques, such as those in [4][5], rely on shape information from other (typically previous) time instants to perform the concealment. These techniques usually achieve better concealment results when the shape does not change much in consecutive time instants.

In terms of texture data concealment, however, no specific techniques are available for object-based video and, therefore, still have to be developed. The existing frame-based techniques (e.g., [6][7][8]) cannot be directly used because they are not able to deal with the new problems created by the fact that the visible texture data no longer corresponds to a simple rectangle of pixels and is now dictated by the shape data. Since the temporal shape concealment technique proposed in [5] can also be used for texture, this paper proposes a spatial error concealment solution for corrupted object-based video texture data. This way, the technique to be proposed here will also be usable for object-based image coding systems.

2. SPATIAL TEXTURE CONCEALMENT FOR OBJECT-BASED VIDEO

To analyze the problems associated with the concealment of texture data in object-based systems, the adopted type of coding solution has to be specified since this will greatly influence the way concealment has to be performed. Here, it is assumed that some kind of block-based technique,

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

such as in the MPEG-4 Visual standard [1], is used to encode the video object planes (VOPs). This way, although video objects can have an arbitrary shape, as shown in Figure 1 (a), they are transmitted in the form of a rectangular bounding box which is divided in 16×16 (macro) blocks[1]. For each 16×16 block, shape data is transmitted, followed by texture data if needed; the texture data is sent as up to four separate 8×8 blocks. The total transmitted shape corresponds to an alpha plane, as shown in Figure 1 (b), which dictates the parts of the transmitted texture, shown in Figure 1 (c), that will be visible. The blocking artifacts in Figure 1 (c) are due to the padding of 8×8 texture blocks, implemented at the encoder to reach a more efficient coding; at the decoder, the correct texture is obtained by 'filtering' with the shape. In terms of bitstream errors, it is considered that they manifest themselves as bursts of consecutive lost 16×16 blocks, which correspond to video packet losses. Additionally, it is also considered that the shape data of corrupted blocks has already been concealed, e.g. using the solution in [2], and only the texture data needs to be recovered.

(a) (b) (c)

Figure 1 – An object and its transmitted components: (a) Visible object; (b) Transmitted shape data; (c) Transmitted texture data

In object-based image and video coding, three types of 16×16 blocks can be identified in the object bounding box (see Figure 2):

- **Opaque blocks** – Blocks for which all the shape data is opaque;
- **Border blocks** – Blocks with some opaque and some transparent shape data;
- **Transparent blocks** – Blocks for which all the shape data is transparent.

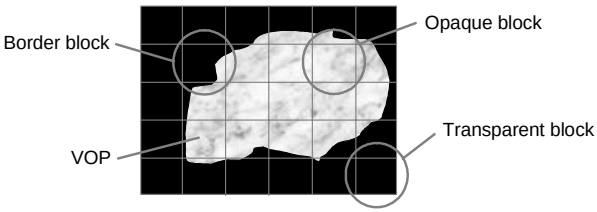


Figure 2 - Types of blocks in a VOP bounding box

From the three block types, only opaque and border corrupted blocks are relevant in terms of texture concealment because they correspond to visible texture. Transparent blocks do not have any texture and, therefore, need no texture concealment. The type of a given block, as well as the pixels that will be visible, is entirely determined by the associated shape data, which for corrupted blocks has also been concealed. Due to shape concealment, the obtained shape may be different from the original, thus influencing the texture concealment results in one of two possible ways:

- **Object shrinks** – Some of the shape data, which originally was opaque, becomes transparent. This means that some corrupted texture for which concealment would be necessary is no longer needed because the corresponding pixels will never be displayed (but texture concealment is still needed for the remaining pixels in the corrupted block).
- **Object expands** – Some of the shape data, which originally was transparent, becomes opaque. This situation is far more complex because texture concealment is now needed for a region of the object where no texture originally existed and therefore has to be made up.

To illustrate these two situations, a corrupted VOP and its shape are shown in Figure 3 (a) and (b). After shape concealment, the alpha plane in Figure 3 (c) is obtained, where the differences

[1] These blocks are 16×16 in the luminance component and 8×8 in the two chrominance components. To save space, explanations are given for the luminance, but also apply to the chrominance with small adjustments.

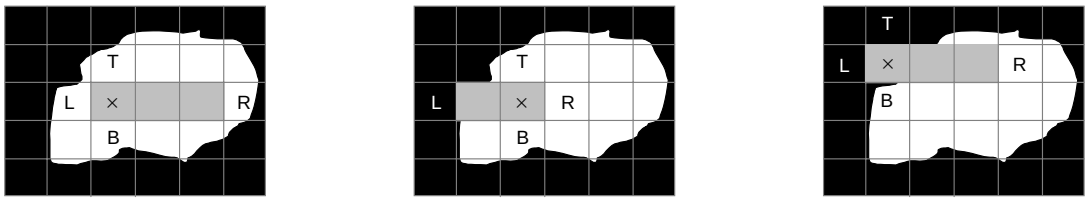
regarding the original (shown in Figure 1) have been highlighted (object shrinking in the left and object expansion at the center/right). Finally, in Figure 3 (d), an example of how this VOP would look with the concealed texture is shown; the technique later described as the weighted median concealment was used here.

(a) (b) (c) (d)

Figure 3 – Illustration of texture concealment problems in object-based coding: (a) Corrupted VOP; (b) Corrupted shape; (c) Concealed shape; (d) Concealed VOP

3. PROPOSED SPATIAL TEXTURE ERROR CONCEALMENT ALGORITHM

To address the problems described above, it is necessary to develop a texture error concealment technique for object-based coding systems. Since the shape data of corrupted blocks is considered to have already been concealed [2] (with possible shrinking or expansion), this technique only has to deal with the texture data in bursts of consecutive lost blocks. To do this, each lost block in a burst is concealed by considering the closest available blocks to each side. Since only correctly received opaque or border blocks can be used, the concealment may have to be done based on less than four blocks. This is illustrated in Figure 4, where the block marked '×' is the block currently being concealed and the closest correctly received blocks to each side are those marked 'T' (top), 'B' (bottom), 'L' (left) and 'R' (right); of the closest correctly received blocks, only those that are opaque or border blocks can be used for concealment.



(a) (b) (c)

Figure 4 – The closest correctly received blocks to each side of the block being concealed: (a) Four blocks can be used; (b) Three blocks can be used; (c) Two blocks can be used

The proposed texture concealment technique includes the following steps:

1. **Padding of correctly received border blocks** – Since the texture of 16×16 border blocks is sent in up to four 8×8 blocks (for which there is some opaque shape data), this means that some parts of these 16×16 blocks may have no texture data. Since these blocks can be used to conceal other corrupted blocks, they should be fully padded in order to make them more useful. Here, padding is done as specified by the MPEG-4 Visual standard [1].
2. **Pixel value estimation for corrupted border and opaque blocks** – In this step, the values of visible pixels in corrupted blocks are estimated based on the closest available blocks to each side. While for opaque blocks this corresponds to all the pixels, for border blocks only the values of some pixels have to be estimated. However, in the case of border blocks, pixel value estimation can still be applied to the whole block because the shape data will later be used to 'filter' the texture that should be visible[2]. In addition, since the closest available blocks to each side are now complete 16×16 texture blocks because border blocks have been padded, in principle it is possible to use at this stage any of the available frame-based texture concealment techniques. However, some

[2] If a faster implementation is desired, the pixel value estimation should only be applied to the pixels that will be visible. Otherwise, resources are being wasted to estimate the values of pixels that will never be visible.

adjustments are needed because only opaque and border blocks can be used for the concealment. To do this, two approaches are proposed here:

- **Linear interpolation** – This approach is based on the frame-based technique proposed in [7], where the missing transform coefficients in a block are recovered by interpolating the corresponding transform coefficients from the spatially adjacent blocks. When all the transform coefficients for a block are lost, as is the case here, the frequency domain interpolation is the same as interpolating each lost pixel in the block from the corresponding pixels in the spatially adjacent blocks. Here, however, the concealment is based on the closest available blocks to each side, which are not necessarily spatially adjacent to the block being concealed. To deal with this, weights are needed to increase or decrease the contributions of pixels depending on their distance. In addition, since the number of closest available blocks to each side can vary (they are at most four), the following cases should be considered[3]:
  - **Four available blocks** – In the case that four blocks are available to recover the visible pixels in a given 16×16 block, the following expression should be used:

$$b_{\times}(i,j) = \frac{1}{d_L + d_R + d_T + d_B} \cdot \left(d_R b_L(i,j) + d_L b_R(i,j) + d_B b_T(i,j) + d_T b_B(i,j) \right) \quad (1)$$

    where b_X is the block being concealed and b_L, b_R, b_T and b_B correspond to the available blocks to each side (i.e., left, right, top and bottom). As for d_L, d_R, d_T and d_B, they correspond to the distances between the centers of block b_X and blocks b_L, b_R, b_T and b_B, respectively. The pair *(i,j)* corresponds to the visible pixel coordinates inside a given block.
  - **Three available blocks or two available blocks on opposite sides** – If only three blocks are available for the concealment, as in Figure 4 (b), the previous expression can no longer be used. Since the fact that no block is available on a particular side affects the weight of the available block on the opposite side, they should both be ignored in Equation 1. For the example in Figure 4 (b), this leads to the following expression:

$$b_{\times}(i,j) = \frac{1}{d_T + d_B} \cdot \left(d_B b_T(i,j) + d_T b_B(i,j) \right). \quad (2)$$

    The same kind of simplification should be done to Equation 1 if only two blocks are available for the concealment and these blocks are on opposite sides of the block being concealed.
  - **Two available blocks not on opposite sides** – If, however, two blocks are available but are not on opposite sides, which is the case in Figure 4 (c), a slight adjustment of the linear interpolation is needed, leading to the expression:

$$b_{\times}(i,j) = \frac{1}{d_R + d_B} \cdot \left(d_B b_R(i,j) + d_R b_B(i,j) \right). \quad (3)$$

    For other combinations of two available blocks that are not on opposite sides, similar expressions could be derived.
  - **One available block** – If only one block is available for the concealment, this block is simply copied to the block being concealed.
- **Weighted median** – In the previous approach, if one of the blocks used for the interpolation is significantly different from the rest, the results can be seriously negatively affected. This can be solved by using the weighted median instead, which has the ability to remove outliers and is defined as:

$$b_{\times}(i,j) = median\left\{ \underbrace{b_L(i,j),\ldots,b_L(i,j)}_{w_L \text{ times}}, \underbrace{b_R(i,j),\ldots,b_R(i,j)}_{w_R \text{ times}}, \underbrace{b_T(i,j),\ldots,b_T(i,)}_{w_T \text{ times}}, \underbrace{b_B(i,j),\ldots,b_B(i,j)}_{w_B \text{ times}} \right\} \quad (4)$$

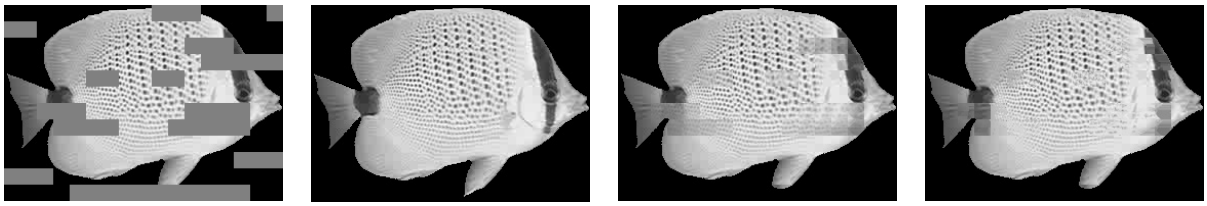
where

$$w_{\times} = \frac{\left(d_L + d_R + d_T + d_B\right) \cdot \max\left\{d_L, d_R, d_T, d_B\right\}}{d_X}. \quad (5)$$

[3] If no blocks are available for the concealment, spatial texture concealment should be aborted and the previous VOP should be simply copied.

The weights w_L, w_R, w_T and w_B are inversely proportional to the distance that separates the block being concealed and the contributing blocks which can be at most four. The numerator guarantees that these constants will be integers, which is necessary because the median can only be computed for an integer number of elements.

To compare the performance of the two proposed concealment approaches, Figure 5 should be considered. In Figure 5 (a) and (b), a corrupted VOP and the corresponding original are shown. In Figure 5 (c) and (d), the same VOP is shown concealed with the two different approaches. As can be seen, when the linear interpolation approach is used, there are several bursts where one darker block negatively influences the concealment of the whole burst; this does not happen when the weighted median is used.



(a) (b) (c) (d)

Figure 5 – A concealment example: (a) Corrupted VOP; (b) Original VOP; (c) Concealed with the linear interpolation; (d) Concealed with the weighted median

4. PERFORMANCE EVALUATION

To evaluate the proposed texture concealment technique, the Akiyo, Bream and Stefan video objects have been encoded according to the MPEG-4 Core Visual Object Type. In terms of error resilience tools, resynchronization markers and data partitioning were used. Since the proposed technique relies only on texture data from surrounding blocks and this can only be done reliably if the corrupted areas are relatively small as typically occurs with the video packet size used in intra coded objects, all the video objects in the sequence have been intra coded.

To simulate channel errors, the decoder simply ignored video packets randomly with a given packet loss rate (following a uniform distribution). For each loss rate, each video object was decoded 50 times (i.e., 50 different error patterns or runs), while applying the shape concealment technique described in [2], which is based on contour interpolation with Bézier curves, and the texture concealment techniques proposed here to the corrupted VOPs.

To evaluate shape quality, the *Dn* metric used by MPEG is adopted; *Dn* is the number of different shapels between decoded and original alpha planes divided by the total number of opaque shapels in the original alpha plane. As for texture quality, it is evaluated with 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 and the original VOP.

In Table 1, the obtained *Dn* values, which are independent of the used texture concealment technique, are shown. 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. As for Table 2 and Table 3, they show the texture concealment results for the linear interpolation and the weighted median, respectively. $PSNR_{low}$, $PSNR_{avg}$ and $PSNR_{high}$ have equivalent definitions to the various *Dn* metrics in Table 1. The average *PSNR* values in the error-free case for the Akiyo, Bream and Stefan video objects are 35.54 dB, 33.20 dB and 32.31 dB, respectively.

As can be seen, the weighted median approach slightly outperforms the linear interpolation approach for the Akiyo and Bream sequences, but not for the Stefan sequence. This is due to the outlier elimination property of the median. Since the Akiyo and Bream video objects have large regions with similar texture, even if one of the blocks used in the concealment has a low correlation with the block being concealed, the final result is not compromised. For Stefan, however, the used blocks are typically very different from each other and from the block being concealed; in this case, the ability to remove outliers has a lower impact.

Table 1 – Dn values for the linear interpolation and the weighted median

Video packet loss rate	Dn [%] (Dn_{low}/Dn_{avg}/Dn_{high})								
	Akiyo			Bream			Stefan		
1%	0.01	0.03	0.04	0.03	0.04	0.09	0.10	0.20	0.33
5%	0.12	0.16	0.25	0.19	0.27	0.40	0.69	1.03	1.70
10%	0.27	0.40	0.55	0.50	0.66	0.83	1.77	2.34	2.88
20%	0.92	1.23	1.54	1.46	1.96	2.55	5.27	6.12	7.08

Table 2 – PSNR values for the linear interpolation

Video packet loss rate	PSNR [dB] ($PSNR_{low}$/$PSNR_{avg}$/$PSNR_{high}$)								
	Akiyo			Bream			Stefan		
1%	34.24	34.50	34.69	32.13	32.29	32.47	31.44	31.63	31.77
5%	31.28	31.61	31.95	29.39	29.63	29.92	29.01	29.41	29.76
10%	28.94	29.28	29.54	27.19	27.47	27.75	26.72	27.25	27.61
20%	26.29	26.49	26.67	24.54	24.77	24.94	23.82	24.36	24.72

Table 3 – PSNR values for the weighted median

Video packet loss rate	PSNR [dB] ($PSNR_{low}$/$PSNR_{avg}$/$PSNR_{high}$)								
	Akiyo			Bream			Stefan		
1%	34.33	34.58	34.74	32.16	32.32	32.50	31.41	31.61	31.78
5%	31.55	31.85	32.18	29.50	29.73	30.03	28.96	29.36	29.68
10%	29.29	29.57	29.72	27.35	27.59	27.85	26.63	27.17	27.58
20%	26.51	26.70	26.90	24.69	24.88	24.99	23.64	24.22	24.61

5. FINAL REMARKS

In this paper, a spatial concealment technique was proposed to conceal texture errors in object-based images and video. This technique, which is the only one available in the literature for object-based systems, consists of two steps. In the first step, padding is applied to the available texture. This is followed by the second step, where the lost pixel values are estimated; in order to use existing frame-based techniques for this, they would have to be modified, namely in terms of the surrounding blocks that are used. For the pixel value estimation step, two different approaches were proposed, for which results have been presented, showing their ability to recover lost texture data in a quite acceptable way.

Finally, it is important to emphasize the relevance of concealment techniques in order to be able to actually deploy object-based video applications in error-prone environments with an acceptable video quality at the decoder.

6. REFERENCES

[1] ISO/IEC 14496-2, "Information Technology – Coding of Audio-Visual Objects, Part 2: Visual," December 1999.

[2] L. D. Soares, F. Pereira, "Spatial Shape Error Concealment for Object-based Image and Video Coding," *IEEE Trans. on Image Proc.*, Vol. 13, No. 4, pp. 586-599, April 2004.

[3] G. M. Schuster, X. Li, A. K. Katsaggelos, "Shape Error Concealment Using Hermite Splines," *IEEE Trans. on Image Proc.*, Vol. 13, No. 6, pp. 808-820, June 2004.

[4] P. Salama, C. Huang, "Error Concealment for Shape Coding," *Proc. ICIP 2002*, Rochester, NY, Vol. 2, pp. 701-704, September 2002.

[5] L. D. Soares, F. Pereira, "Motion-based Shape Error Concealment for Object-based Video," *Proc. ICIP 2004*, Singapore, October 2004.

[6] Y. Wang, Q.-F. Zhu, L. Shaw, "Maximally Smooth Image Recovery in Transform Coding," *IEEE Trans. on Comm.*, Vol. 41, No. 10, pp. 1544-1551, October 1993.

[7] S. Hemami, T. Meng, "Transform Coded Image Reconstruction Exploiting Interblock Correlation," *IEEE Trans. on Image Proc.*, Vol. 4, No. 7, pp. 1023-1027, July 1995.

[8] S. Aign, K. Fazel, "Temporal & Spatial Error Concealment Techniques for Hierarchical MPEG-2 Video Codec," *Proc. ICC 95*, Vol. 3, pp. 1778-1783, Seattle, WA, November 1995.