Temporal Shape Error Concealment by Global Motion Compensation With Local Refinement

Luis Ducla Soares, *Member, IEEE,* and Fernando Pereira, *Senior Member, IEEE*

***Abstract*—This paper presents an original temporal shape error concealment technique based on a combination of global and local motion compensation. For this technique, which is especially useful for object-based video applications in error-prone environments (e.g., mobile networks), 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 first assumes that a global motion model can describe the shape changes in consecutive time instants. This way, based on locally estimated global motion parameters, the decoder attempts to conceal the corrupted alpha plane by global motion compensating the shape data from the previous time instant. Afterwards, since a global motion model cannot perfectly describe all alpha plane changes, a local motion refinement is applied to improve the concealment in areas of the object with significant local motion.**

***Index Terms*—Error concealment, global motion compensation, local motion refinement, object-based video, shape concealment.**

I. 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; this approach may have both advantages in terms of coding efficiency as well as in terms of additional functionalities. 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 work for object-based video, shape concealment schemes are just starting to emerge. Nevertheless, the problem of shape data concealment has already been addressed by some researchers and several techniques have already been proposed in the literature. Depending on what information is used for the concealment, these techniques can be divided in two different categories, each one with its own advantages and disadvantages. On one hand, spatial error concealment techniques, such as those proposed in [2]–[7], only use spatially adjacent shape information to perform the concealment and do not use any shape information from other time instants. This makes these techniques especially useful when the shape changes greatly in consecutive time instants, such as in scene changes (where there are in fact new objects) or when objects are uncovered. Additionally, they can also be used for still images. On the other hand, temporal error concealment techniques, such as those proposed in [8] and [9], rely on shape information from other (typically previous) time instants to perform the concealment. Since, in most video object sequences, the shape data does not change that much in consecutive time instants, these techniques are typically able to achieve better concealment results. Ideally, the decoder should have access to both types of techniques and choose between them or combine them, according to the situation at hand.

Since this paper focuses on temporal shape error concealment, which has been addressed before but where a lot of improvements can still be made, it is important to review the two major temporal shape error concealment techniques available in the literature (i.e., those proposed in [8] and [9]), as well as their limitations. In the first of these techniques [8], the idea of motion-compensated concealment, which is commonly used for block-based texture, is extended to block-based shape data. This way, when a given block of shape data is corrupted, the decoder tries to conceal it by copying a block of shape data from the previous time instant. To do this, three different (and increasingly complex) approaches are proposed. The first approach is simply to copy the co-located block of shape data from the previous time instant. The second approach consists in using the motion vector associated with the block located above the corrupted block, in the current time instant, to find a replacement shape block for the corrupted block from the previous time instant. Finally, the third approach consists in using the correctly decoded shape data blocks surrounding the corrupted block to estimate a motion vector for the corrupted shape block, which can then be used to find the replacement block from the previous shape data. This latter approach represents an adaptation to shape data of the technique described in [10], which only targets the concealment of corrupted texture blocks in frame-based video. This is performed by using the correctly decoded texture data surrounding the corrupted block to estimate a motion vector for the corrupted texture block, which is then used to find the replacement block from the previous texture data. However, it is important to notice that this problem is fundamentally different from what is considered here.

The main drawback of the technique described in [8], however, is that the concealment is performed block by block, based on information from neighboring blocks (which may have been

Manuscript received March 23, 2004; revised June 7, 2005. The work presented was developed within VISNET, a European Network of Excellence (http://www.visnet-noe.org), supported by the European Commission through the IST FP6 Program. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Charles D. "Chuck" Creusere.

The authors are with Instituto Superior Técnico—Instituto de Telecomunicações, 1049-001 Lisboa, Portugal (e-mail: lds@lx.it.pt, fp@lx.it.pt).

Digital Object Identifier 10.1109/TIP.2006.871167

concealed themselves), instead of considering the whole shape. This means that it typically only provides good results for isolated lost blocks. When several neighboring blocks are lost and need to be concealed (as it typically happens in practical situations), the concealment performance degrades very fast.

Later, the method proposed in [9] eliminated this problem by considering the whole shape data, instead of using the block-based approach to concealment. In [9], the proposed method is based on the idea that a global motion model can be used to describe the contour changes (i.e., the boundary of the shape data) of a given video object, in consecutive time instants. 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.

However, this method also has problems associated with it. Its first drawback is that the global motion parameters are computed at the encoder, when the sequence is being encoded. These parameters are then transmitted to the decoder side using a separate stream, which according to the authors of [9] represents a 5% bit rate increase. This can be a serious limitation because the technique can then 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 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 technique proposed in [9] 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.

In this paper, an original temporal shape error concealment technique will be presented. This technique, which relies on a combination of global and local motion compensation, and provides significant improvements when compared to the techniques already available in the literature, is described in Section II. Results showing its performance under relevant conditions are included in Section III. In Section IV, some final remarks will be made.

II. Proposed Temporal Shape Error Concealment Technique

In this proposal, the idea of using global motion to conceal shape errors at the decoder is extended while eliminating the two major drawbacks mentioned above. In the technique here proposed, all the processing is performed at the decoder, which no longer has to rely on additional information sent by the encoder. This removes the need to normatively define the concealment information to be sent by the encoder (the encoder has now nothing to send) since the proposed technique can 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 in the context of a specific coding standard since concealment techniques are not normative. This way, the additional bit rate that would be used by the encoder to send the extra stream can be used for something else, such as for 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 shape alpha planes have been encoded with some kind of block-based technique before being delivered, such as (but not necessarily) the MPEG-4 Visual standard [1]. It is also considered that bitstream errors will manifest themselves as bursts of consecutive erroneous 16×16 blocks, which have to be detected by an error detection stage before applying the concealment itself.

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

After the global motion parameters have been determined, they can be used to global motion compensate the alpha plane of the previous time instant. The obtained global motion compensated alpha plane is considered as the reference alpha plane and it will be used to perform the concealment. 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 reference alpha plane. Assuming that the global motion model can accurately describe the shape motion, this alone should be able to produce very good results. However, in many cases, the shape motion cannot be accurately described only by global motion, due to the existence of local motion in some areas of the (non-rigid) shape. Therefore, to avoid significant differences when concealing erroneous blocks in areas with local motion, an additional local motion refinement scheme has been introduced. 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 to be used to find a better replacement block in the previous alpha plane for the erroneous shape block is estimated (in this case, the global motion compensation will not be used, which means that the local motion supersedes the global motion). As a result of this whole concealment process, it is possible that small regions appear in the concealed alpha plane, which can have a very negative impact on the user. Therefore, in order to remove those regions, a final postprocessing step has also been introduced.

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

In order to understand better the shape concealment process, an illustrative example will be given before detailing each of

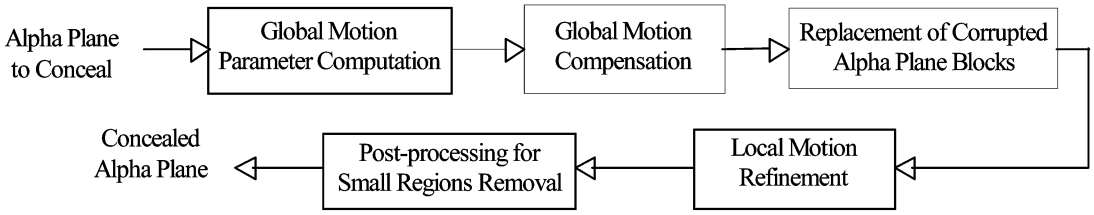

Fig. 1. Proposed temporal shape concealment process.

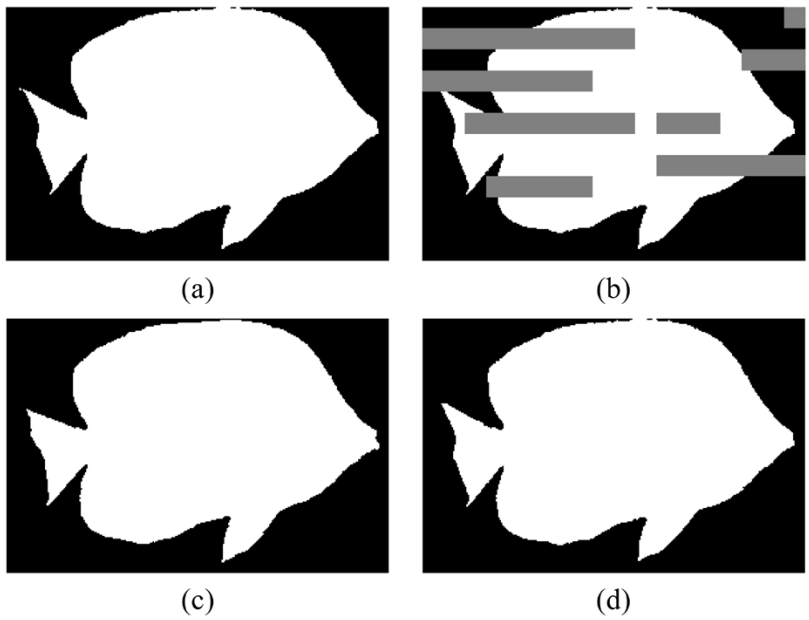

Fig. 2. Exemplifying the steps of the temporal shape concealment process for the Bream video object (CIF, 10 fps). (a) Original uncorrupted alpha plane; (b) corrupted alpha plane; (c) motion compensated previous alpha plane; (d) concealed alpha plane without local motion refinement.

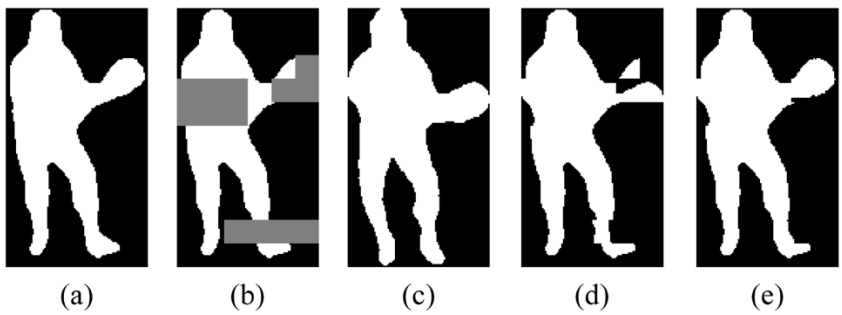

Fig. 3. Exemplifying the steps of the temporal shape concealment process for the Stefan video object (CIF, 15 fps). (a) Original uncorrupted alpha plane; (b) corrupted alpha plane; (c) motion compensated previous alpha plane; (d) concealed alpha plane without local motion refinement; (e) concealed alpha plane with local motion refinement.

the steps above in Sections II-A–II-E. In this example, the alpha plane in Fig. 2(a) has been corrupted, as illustrated in Fig. 2(b). Based on the estimated global motion parameters [using the information in Fig. 2(b)], the previous video object plane (VOP) is motion compensated and Fig. 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 Fig. 2(d). In this case, since the shape and its motion are very simple, the global motion model is able to describe the shape motion accurately enough, and thus no refinement would be needed for local motion. However this is not always the case, notably for humans.

In some other sequences, the objects may have a lot of local motion in addition to the global motion, as is the case in Fig. 3. As can be seen from Fig. 3(d), by simply replacing the corrupted shape blocks with the co-located blocks from the global motion compensated previous alpha plane, this would lead to very annoying artifacts. Therefore, to avoid this, an additional refinement step is needed to deal with the local motion. The results with this refinement are shown in Fig. 3(e), which no longer exhibits the most annoying artifacts.

At first, one might be tempted to think that the poor concealment results achieved in Fig. 3(d) are due to the fact that the global motion parameters have been computed based on incom-

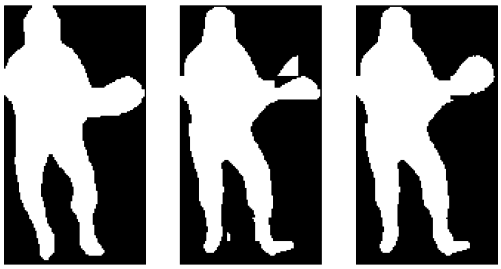

Fig. 4. Same temporal shape concealment process as Fig. 3, but with global motion parameters determined with the complete uncorrupted shape data. (a) Motion-compensated previous alpha plane; (b) concealed alpha plane without local motion refinement; (c) concealed alpha plane with local motion refinement.

plete shape data and, thus, are inaccurate. However, this is not the case, as can be clearly seen in Fig. 4, where exactly the same concealment has been done, but the used global motion parameters have been previously determined based on the complete uncorrupted shape data. As can be seen in Fig. 4(b), the obtained concealment results are equally poor; this is simply due to the fact that local motion exists, which cannot be described by the global motion model.

Additionally, as mentioned above, it is possible for small regions to appear in the concealed alpha plane as a result of the concealment process. Since their removal is a conceptually very simple operation and an illustrative example will be given in Section II-E, this operation is not illustrated here. As shall be explained in Section II-E, since this operation involves the risk of actually eliminating important information, this filter can always be turned off if the user should so wish.

A. Global Motion Parameters Computation

Before computing the global motion parameters, a global motion model has to be chosen from those available in the literature. The differences between the existing global motion models are basically related to their complexity and, thus, their capacity to describe more complicated motion trajectories (of rigid objects). Since the global motion model will be used to describe the motion of the shape data from one time instant to the next, which is typically quite simple, a simple model should be enough. However, it is always possible to replace the used global 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 would eventually allow more sophisticated decoders to have more powerful error concealment capabilities at the cost of some additional computational power.

Taking into account the considerations above, the affine four parameter model, which is certainly the simplest and most widely used global motion model, was adopted. This simple model is adequate for most motion trajectories and its choice should facilitate the deployment of this concealment technique by contributing as little as possible to increase the complexity of the decoder. A description of how this model is derived can be found in [11]. The types of motion that is possible to adequately describe with this motion model are: change of the camera focal length (i.e., zoom or scale), rotation around an axis normal to the camera axis (i.e., pan or tilt) and rotation around the camera axis.

With the selected affine model, when two time instants are considered, as well as forward motion, 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 using (1)

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

where c_1, c_2, c_3, and c_4 are the affine model global motion parameters. Parameter c_1 is associated with *zoom*, parameter c_2 with *rotation* and parameters c_3 and c_4 with *pan* and *tilt*, respectively.

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 global motion basically corresponds to the motion of the contour. Thus, to determine the global motion of the shape data, the following three steps have to be performed.

1) Extraction of the Available Video Object Contour: As mentioned above, 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. Here, as shown in Fig. 5, the shapel representation of the contour will be used. To extract the contour from the corrupted binary alpha plane and represent it using a shapel-based approach, a very simple contour filter has to be implemented. For each available opaque shapel in the corrupted alpha plane, the 4 closest neighbors (considering 4-connectivity) are examined: if at least one of them is transparent, the opaque shapel is considered a contour point. With this filter, the obtained (corrupted) contour is 8-connected.

2) Determination of the Motion Vectors Associated With a Set of Contour Points: The next step is to determine, for a set of selected points of the extracted contour (or eventually all the points), the corresponding points in the previous alpha plane; this is the same as determining a motion vector for each of the considered contour points. Here, to guarantee the best possible global motion estimation, all the contour points in the available contour will be considered. However, in situations where the computational complexity is an issue, less contour points can be used, as will be discussed later. This way, for each of the considered contour points, a block of shapels from the current alpha plane, centered on the contour point coordinates, is considered; this shape block will be referred to in the following as the *shape context* of that contour point. Then, with the shape context, a search is carried out to find (within a given search range) a similar shape block in displaced locations of the previous alpha plane. After the most similar shape block has been found, the motion vector associated with the considered contour point will be simply given by the difference between the position of the found block in the previous alpha plane and the position of the shape context in the current alpha plane. As for the point in the previous alpha plane corresponding to the considered contour point, it is simply the center of the found shape block.

Fig. 5. Bream video object. (a) Alpha plane; (b) shapel representation of the contour.

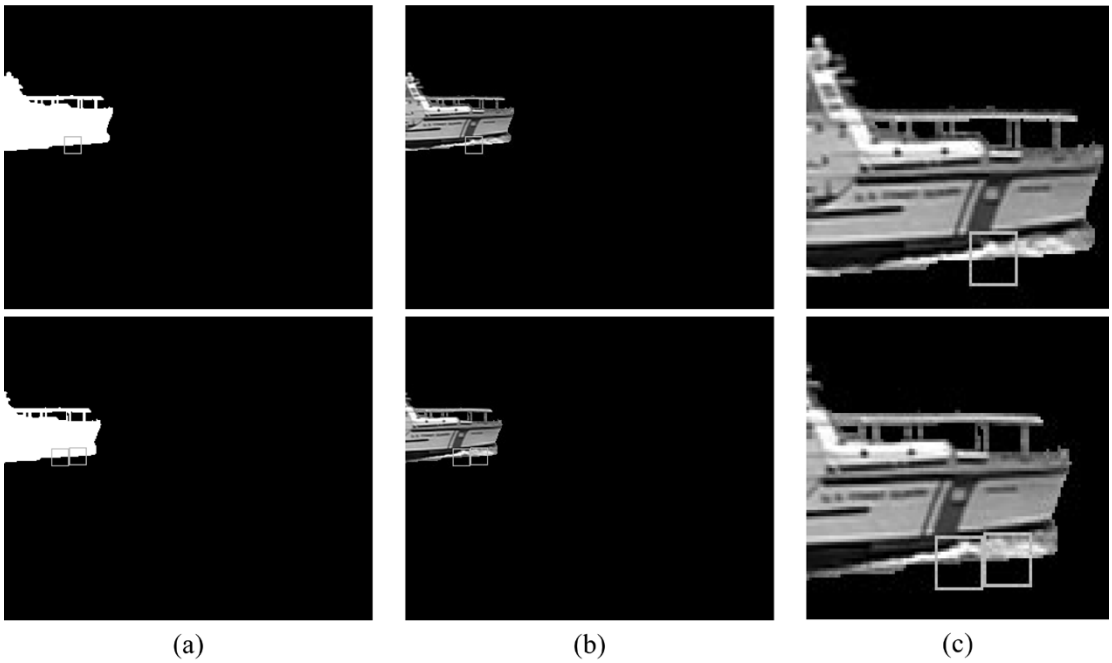

Fig. 6. Determination of contour point motion vectors. (a) Current alpha plane with the shape context of the contour point under consideration (top) and previous alpha plane with two shape blocks perfectly matching the considered shape context (bottom); (b) current texture with the texture context of the contour point under consideration (top) and previous texture with the two texture blocks whose corresponding shape perfectly matches the considered shape context (bottom); (c) detail of (b).

This search for a shape block in the previous alpha plane similar to the shape context can be quite a complex task because, in many cases, several perfectly matching candidates can be found, some of them leading to disastrous results in terms of global motion parameters. Therefore, to improve the shape global motion estimation, the texture data will also be used. Thus, the same procedure described above is followed, but instead of using the shape context of the considered contour point, a *texture context* is used. The texture context is simply a block of pixels from the current luminance plane, centered on the contour point coordinates. This texture context is used to carry out the search in the previous luminance plane to find a similar texture block.

Here, the used texture context is a block of 16×16 pixels and the search range is 32 pixels in total (16 to each side); however, these values can easily be changed. To measure the similarity between texture blocks, the sum of absolute differences (SAD) is used, as it is typically done in motion estimation algorithms. To favor the accuracy of the estimation, an exhaustive search pattern is adopted. However, a faster, although sub-optimal, algorithm such as three step hierarchical search [12] could be selected to decrease the computational complexity associated with this processing step.

To illustrate how the use of texture can improve the motion estimation results, the Large Boat object of the Coastguard sequence (CIF, 10 fps) is considered. At the top of Fig. 6(a), the shape context of the contour point with coordinates (65, 132) is shown. If only the alpha plane is used to determine the motion vector associated with this contour point, two ambiguous shape blocks appear within the search range as illustrated at the bottom of Fig. 6(a), which perfectly match the shape context: one centered on coordinates (53, 133) and another centered on coordinates (70, 132). However, when the texture information of the object is used, no ambiguity occurs. At the top of Fig. 6(b), the texture context of the contour point with coordinates (65, 132) is shown. As can be seen at the bottom of Fig. 6(b), of the two previously found shape blocks, one of them has a texture more similar to the texture context of the considered contour point and, therefore, will be chosen as the matching point. This is shown with further detail in Fig. 6(c).

Of all the steps in the proposed shape concealment technique, this one is by far the most computationally complex and it will basically dictate the computational complexity of the whole technique. Since this step involves the determination of one motion vector for several contour points in the available contour, its complexity basically depends on 1) the number of contour points for which a motion vector has to be determined, 2) the search range used to determine the motion vector, and 3) the type of search algorithm used to find the motion vector (exhaustive or not). These three dimensions make the technique intrinsically scalable in terms of computational complexity because the decoder can easily adapt it to its processing power. This basically means that the computational complexity of this technique will largely depend on what the decoder decides to do. For instance, while very powerful decoders may choose to determine one motion vector for each available contour point, less powerful decoders may decide to determine motion vectors for only some of the contour points (e.g., one out of five or even ten points or even for some specific, more relevant points). Since this leads to a very large complexity reduction without significantly affecting the global motion parameter values, it can be a good way for the decoder to adapt the concealment technique to its processing power. As an example, if a 500-point contour is considered, the computational complexity can indeed be high if a motion vector is computed for each one of the 500 points. However, the decoder may choose to use only one out of ten points, meaning that only 50 motion vectors will have to be computed and that the computational complexity will be reduced tenfold. This complexity, however, can still be further reduced by carefully choosing only the most relevant points within the 50. Another way for the decoder to adapt the technique to its computational power is by varying the search range used to determine the motion vectors. Although this can be a very effective method to reduce the complexity, it can have rather severe consequences in terms of the estimated global motion parameters if the search range is too small for a given video object sequence and, therefore, should be adopted with care. Finally, as mentioned above, the exhaustive search pattern used here can also be replaced by a faster nonexhaustive one, such as the three-step hierarchical search [12]. This way, by playing with these three degrees of freedom, a given decoder can adapt the concealment technique to its processing power in order to provide real time performance, if necessary. However, during a typical decoding session, the search algorithm used to estimate the motion vectors will most likely remain the same, as well as the search range (chosen for the type of content that is being decoded). This means that, for a given contour with n contour points, the decoder will still be able to decide to use only 1 out of m such contour points. In this case, the complexity of this step will be $O(n/m)$.

3) Determination of the Global Motion Parameters: After corresponding points in the previous alpha plane have been found for all the selected points in the contour extracted from the current alpha plane, the global motion parameters are determined by finding linear least squares estimates of c_1, c_2, c_3 and c_4. This is done by fitting the model shown in (1) to the N pairs of coordinates obtained in the previous step, which can be accomplished by minimizing the following error expression:

$$E = \frac{1}{N}\sum_{i=1}^{N}\Big[\left(x_i' - (c_1 x_i + c_2 y_i + c_3)\right)^2 + \left(y_i' - (-c_2 x_i + c_1 y_i + c_4)\right)^2\Big] \quad (2)$$

where (x_i', y_i') are the coordinates of the ith contour point extracted from the current alpha plane, and (x_i, y_i) are the coordinates of the corresponding point in the previous alpha plane.

To find the minimum for the error expressed in (2), it is simply a question of computing its derivatives with respect to the four global motion parameters (i.e., c_1, c_2, c_3, and c_4) and then setting them to zero, as shown in:

$$\left[\frac{\partial E}{\partial c_k} = 0\right]_{k=1,2,3,4}. \quad (3)$$

This leads to the four equations that make up the following easy to solve linear system of equations, which will give the values of c_1, c_2, c_3, and c_4, as shown in (4) at the bottom of the page. To solve this equation system, the singular value decomposition (SVD) method, the method of choice for solving most linear least squares problems, which is detailed in [13], was used.[1] This method is not new and, therefore, it will not be detailed here.

A possible problem associated with the computation of global motion parameters is the undesired bias to the results caused by either local motion or erroneously determined matching contour point pairs. To diminish this problem, an iterative outlier removal approach is used. This way, after the first set of motion

[1]If the SVD method does not converge after 30 iterations, it is considered that the global motion parameters cannot be determined (as suggested in [13]); in this case, the corrupted VOP is simply replaced by the one from the previous time instant.

$$\begin{bmatrix} \sum_{i=1}^{N}(x_i^2+y_i^2) & 0 & \sum_{i=1}^{N} x_i & \sum_{i=1}^{N} y_i \\ 0 & \sum_{i=1}^{N}(x_i^2+y_i^2) & \sum_{i=1}^{N} y_i & -\sum_{i=1}^{N} x_i \\ \sum_{i=1}^{N} x_i & \sum_{i=1}^{N} y_i & N & 0 \\ \sum_{i=1}^{N} y_i & -\sum_{i=1}^{N} x_i & 0 & N \end{bmatrix} \cdot \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{N}(x_i' x_i + y_i' y_i) \\ \sum_{i=1}^{N}(x_i' y_i - y_i' x_i) \\ \sum_{i=1}^{N} x_i' \\ \sum_{i=1}^{N} y_i' \end{bmatrix} \quad (4)$$

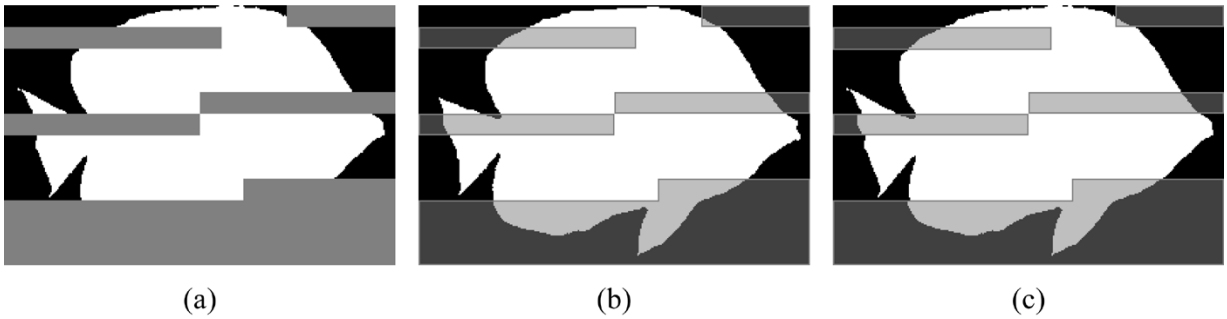

Fig. 7. Replacement of corrupted alpha plane blocks. (a) Corrupted alpha plane; (b) previous global motion compensated alpha plane; (c) concealed alpha plane.

parameters c_1, c_2, c_3, and c_4 has been determined, the residual squared motion compensation error is computed for all the available matching point pairs. This residual error can be determined with the following expression:

$$E_r = (x'_i - (c_1 x_i + c_2 y_i + c_3))^2 + (y'_i - (-c_2 x_i + c_1 y_i + c_4))^2 \quad (5)$$

which corresponds to the squared distance that separates the contour point with coordinates (x'_i, y'_i) extracted from the current alpha plane and the corresponding point in the previous alpha plane with coordinates (x_i, y_i) after it has been motion compensated with the determined motion parameters. All the contour point pairs, whose residual squared error is more than a standard deviation away from the mean squared error, given by (2), will be removed from the set of contour points used to determine the global motion parameters. With the new reduced set of contour matching point pairs, new global motion parameters are computed. Then, the outlier removal procedure is repeated until the global motion parameters converge. The global motion parameters are considered to have converged when they do not change substantially from one iteration to the next (i.e., the change in the global motion parameters is so small that the corresponding changes in terms of the motion compensated results are smaller than a pixel and, hence, not visible). To avoid determining unreliable global motion parameters, this procedure is stopped if the number of contour matching point pairs drops below an empirically determined value of 15; in this case, it is considered that no global motion exists and the global motion parameters are set accordingly (i.e., $c_1 = 1$, $c_2 = 0$, $c_3 = 0$, and $c_4 = 0$), which is basically equivalent to simply copying the alpha plane from the previous time instant.

B. Global Motion Compensation

After the global motion parameters are known, the global motion compensation itself is quite straightforward. To do this, the decoder has to 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 (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.

C. Replacement of Corrupted Alpha Plane Blocks

After the previous alpha plane has been (global) motion compensated, the concealment itself can start. This way, in the corrupted alpha plane, all the blocks that were considered corrupted are replaced with the corresponding alpha plane blocks from the global motion compensated previous alpha plane. The same can be done for the texture data.

To better understand this procedure, Fig. 7 should be considered. In Fig. 7(a), the corrupted alpha plane that is going to be concealed is shown. The gray areas correspond to the corrupted alpha blocks that have to be concealed. In Fig. 7(b), the previous global motion compensated alpha plane is shown, where the shaded areas are the alpha blocks that correspond (i.e., have the same position) to the corrupted alpha blocks in the corrupted alpha plane. Therefore, these blocks simply have to be copied to the corrupted alpha plane. The result is shown in Fig. 7(c), where the concealed blocks are shaded. Notice that the proposed method is able to deal with long bursts of corrupted blocks and not just isolated corrupted blocks as some techniques proposed in the literature.

D. Local Motion Refinement

The case illustrated in the previous section represents a simple situation where practically no local motion exists and, therefore, nothing else would probably have to be done to improve the shape concealment results. However, there are cases, such as the one illustrated back in Fig. 3, where important local motion exists and thus global motion compensation alone is not enough to produce good results. To deal with these more complex cases, where significant local motion exists, a concealment refinement is necessary. However, before trying to refine the concealment, it is important to distinguish two types of concealed blocks:

- blocks that have at least one correctly (shape) decoded surrounding block among their eight closest neighbors;
- blocks that only have surrounding blocks (among their eight closest neighbors) that have been (shape) concealed themselves.

Since the former have much more reliable surrounding shape data (because it has been correctly decoded), the refinement should start with those. The concealment refinement is applied to the latter blocks only afterwards. Thus, the refinement procedure consists of the two consecutive steps described next.

1) Concealment Refinement of Shape Blocks With Correct Neighbors: To refine the concealment of blocks that have at

least one correctly decoded neighboring block among their eight closest neighbors, 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 following set of successive operations has to be performed.

1) **Determination of the need for local motion refinement**—The first step allows to determine if the considered shape block has been acceptably concealed with the global motion compensation alone or not. This is done by inspecting the surrounding correctly decoded 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 surrounding correctly decoded shape blocks and 30 for any individual block, the shape block at hand is considered to have been adequately concealed and thus no further processing is needed for it. These thresholds have been determined by exhaustive experimentation and were chosen because they lead to the best results.
2) **Determination of local motion vectors for the surrounding shape blocks**—If the considered shape block is considered not adequately concealed, the local motion concealment refinement has to be applied. For this, a motion vector is determined for each one of the nontransparent 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. The search range used to determine the motion vectors is the same that was used in Section II-A (i.e., 32 pixels in total, 16 on each side).
3) **Determination of an average local motion vector for the shape block being refined**—Based on the motion vectors determined for all the nontransparent correctly decoded neighboring shape blocks in Step 2), an average motion vector can be determined for the shape block whose concealment is being refined. In order to guarantee that the components of this vector are integers, rounding to the nearest integer is used. 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 computations of the average motion vector.
4) **Refinement by local motion compensation**—After the average motion vector has been determined in Step 3), motion compensation concealment is tried for the shape block at hand with several motion vectors (i.e., replacing the corrupted shape block with the shape block from the previous alpha plane indicated by the considered concealment motion vector). The average motion vector determined in Step 3) is tested, as well as the motion vectors from the shape blocks above, below, to the left and to the right. Of course, the four last motion vectors are only tried if the corresponding shape blocks exist and were either correctly decoded or already (concealed and) refined. From the candidate motion vectors, the one that will be used to perform the final concealment is the one that, when used to find a replacement in the previous alpha plane for the corrupted shape block, minimizes a shape border continuity metric in the concealed alpha plane. This metric is computed by considering the shapels in the four borders of the shape block copied from the previous alpha plane using the various candidate motion vectors and the shapels across these borders in the concealed alpha plane. For an $N \times N$ block (here N equals 16), this shape continuity metric (SCM) is defined as

$$\mathrm{SCM} = \sum_{i=1}^{N} \left|t_i - \hat{t}_i\right| + \sum_{i=1}^{N} \left|b_i - \hat{b}_i\right| + \sum_{i=1}^{N} \left|l_i - \hat{l}_i\right| + \sum_{i=1}^{N} \left|r_i - \hat{r}_i\right| \quad (6)$$

where t_i, b_i, l_i, and r_i correspond to shapels in the top border, the bottom border, the left border and the right border of the shape block copied from the previous alpha plane using the various candidate motion vectors, respectively. As for $\hat{t}_i$, $\hat{b}_i$, $\hat{l}_i$, and $\hat{r}_i$, they correspond to the shapels in the concealed alpha plane across the border from t_i, b_i, l_i, and r_i, respectively. To compute the SCM, only the neighboring blocks that have either been correctly decoded or already undergone local refinement are taken into account. In Fig. 8, this metric is illustrated for an 8×8 block. The same motion vector can also be used to refine the concealed texture data.

Before proceeding to the next step of the refinement, Fig. 9 should be considered since it illustrates well the advantages of this first step. In Fig. 9(a), the corrupted alpha plane that is going to be concealed is shown, where all the corrupted shape blocks (shown in gray) have at least one correctly decoded neighbor. In Fig. 9(b), the global motion compensated previous alpha plane is shown. To help the reader, in Fig. 9(b), as well as in the remainder of Fig. 9, the position of the corrupted shape blocks will be shaded. As for Fig. 9(c), it shows the concealed alpha plane before the local motion refinement has been applied, while Fig. 9(d) shows it after it has been applied. As can be seen from Fig. 9(c), when the corrupted shape blocks are concealed by simply copying co-located shape blocks from the previous (global) motion compensated alpha plane, only the areas of the object where hardly any local motion exists are adequately dealt with. However, for the areas where local motion exists, such as the area of the racket, the concealment results are very poor, especially when compared to what was achieved in Fig. 9(d). To compare the concealed alpha planes with the original uncorrupted alpha plane, the reader can refer back to Fig. 3(a).

2) Concealment Refinement of Shape Blocks With No Uncorrupted Neighbors: To refine the concealment of the shape blocks that do not have any uncorrupted decoded neighbors, 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

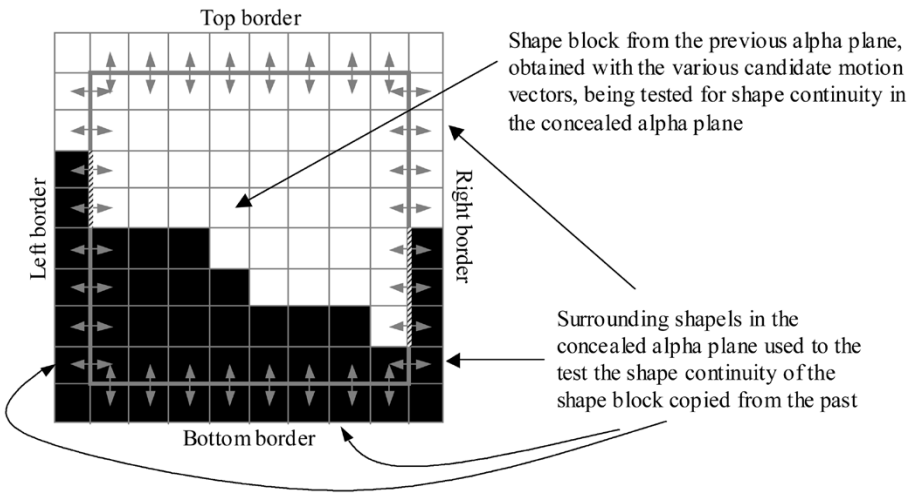

Fig. 8. Example of continuity metric computation; the borders across which there are different shapels are shown with a dashed line (this happens for two shapels in the left border and for three in the right border).

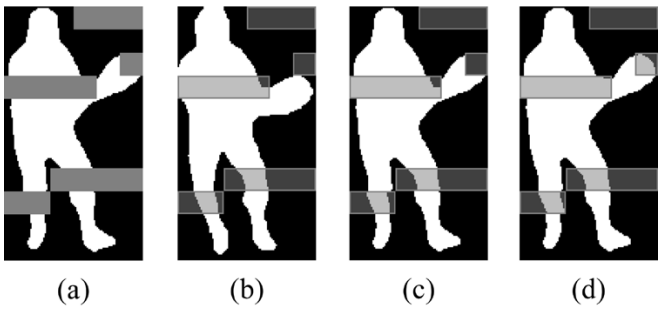

Fig. 9. Replacement of corrupted alpha plane blocks. (a) Corrupted alpha plane; (b) global motion compensated previous alpha plane; (c) concealed alpha plane with global motion compensation only; (d) concealed alpha plane after shape blocks with correctly decoded neighbors have been local motion refined.

blocks, the following set of successive operations has to be performed.

1) **Determination of the existence of locally refined neighbors**—The first step is to determine if any of the surrounding shape blocks have been locally refined or not. If not, nothing is done at this time and no further processing is needed for this shape block.
2) **Determination of an average local motion vector for the shape block being refined**—If, on the other hand, some of the surrounding shape blocks have already been locally refined, their previously determined (i.e., in the previous section) motion vectors should be considered and used to compute an average motion vector. As in the first step of Section II-D, the components of this vector are rounded to the nearest integers.
3) **Refinement by local motion compensation**—Afterwards, motion compensation concealment is tried for the shape block at hand with several motion vectors, as in the first step of . The average motion vector determined in step 2 is tested, as well as the motion vectors from the shape blocks above, below, to the left and to the right. Of course, the four last motion vectors are only tried if the corresponding shape blocks exist and were already (concealed and) local motion refined. From the tested motion vectors, the one that will be used to perform the final concealment is the one that minimizes the shape border continuity metric, defined in (6), for the concealed shape block. 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 set of operations described above has to be performed again from bottom to top, right to left. To better understand why, suppose that the first three rows of blocks in an alpha plane have been corrupted. After the shape blocks in the third row, which are the only ones that have correctly decoded neighbors, have been refined as described in the previous section, the first two rows of corrupted shape blocks still remain to be refined. At this point, only the second row of corrupted shape blocks has refined neighbors. Since the concealment refinement proposed in this section depends on the existence of refined neighbors, only the second row will be refined when performing the described set of operations from top to bottom, left to right. As for the first row of corrupted shape blocks it will only have refined neighbors after the second row has been refined and, therefore, it will only be refined when the processing is being done from bottom to top, right to left.

To illustrate the advantages of this second step of the concealment refinement, Fig. 10 should be considered. Fig. 10(a) shows a corrupted alpha plane, where the corrupted shape blocks are shown in gray. In this case, not all the corrupted shape blocks have at least one correctly decoded neighbor; there is one shape block with no correctly decoded neighbors, which has been indicated in lighter gray. As for Fig. 10(b), it shows the global motion compensated previous alpha plane, where the position of the corrupted shape blocks is shaded. Fig. 10(c) shows the concealed alpha plane before any local motion refinement has been done, while Fig. 10(d) and (e) shows the concealed alpha plane after the first and the second steps of the local refinement, respectively. As can be seen, the concealment results are gradually improved, approaching the uncorrupted original. For the uncorrupted alpha plane, the reader can refer back to Fig. 3(a).

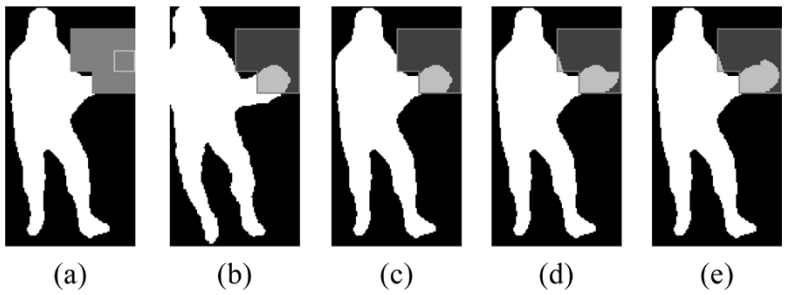

Fig. 10. Replacement of corrupted alpha plane blocks. (a) Corrupted alpha plane; (b) previous global motion compensated alpha plane; (c) concealed alpha plane with global motion compensation only; (d) concealed alpha plane after the shape blocks with correctly decoded neighbors have been refined; (e) concealed alpha plane after all the shape blocks have been refined.

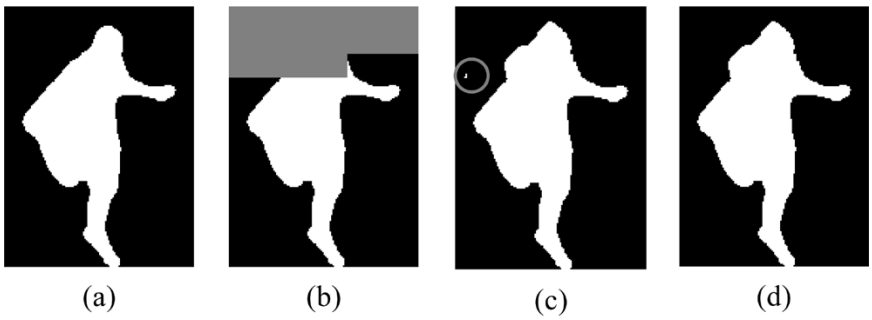

Fig. 11. Elimination of small opaque region inside the concealed area but in contact with its border. (a) Original uncorrupted alpha plane; (b) corrupted alpha plane; (c) concealed alpha plane before small regions elimination; (d) concealed alpha plane after small regions elimination.

E. Postprocessing for Small Regions Removal

Sometimes, after the motion concealment operations are over (i.e., global motion compensation concealment followed by local motion refinement), small opaque or transparent regions appear in the concealed alpha plane. Although these regions are typically very small and, therefore, have a very small influence on the objective quality metrics, their subjective impact can be quite negative. Thus, it is important that these small regions be dealt with. The following two types of regions have been identified.

- **Small opaque or transparent regions inside the concealed area but in contact with its border**—These small regions, located inside the concealed area, but in contact with its border, are a direct consequence of the concealment operations themselves, in the sense that the shapels of these regions have been "created" by the concealment. After all, during concealment, it is perfectly possible for a group of shapels to be erroneously concealed in such a way as to create an isolated region inside the concealed area, but in contact with its border. By removing this artificially looking region, the objective quality will be slightly improved, and the impact on the subjective quality will be much better. This is clearly illustrated in Fig. 11.
- **Small opaque or transparent regions outside but in contact with the concealed area**—These other small regions, located outside but in contact with the concealed area, are an indirect consequence of the concealment operations, in the sense that the shapels in these regions have not been "created" by the concealment. After all, since these shapels are outside the concealed areas, they correspond to correctly decoded shapels that have not even been concealed. In fact, these regions are due to the way nearby shapels inside the concealed area have been concealed. Nevertheless, they can create a very negative subjective impact on the user and, therefore, should be eliminated. Of course, by eliminating them, the objective quality of the alpha plane will be slightly worsened. This is clearly illustrated in Fig. 12.

To eliminate these regions, a rather conservative approach was followed due to the risk associated with this operation; after all, there is a risk of actually removing important information. Therefore, only regions smaller than 25 shapels were eliminated. Of course, it is still always possible to turn off this filter if the user should so wish.

To illustrate the elimination of the small regions described above, two examples can be considered. In Fig. 11, the elimination of a small region inside the concealed area, but in contact with its border, is shown, while in Fig. 12 the eliminated region is outside but in contact with the concealed area. In these examples, the concealed shape blocks have not be shaded, as in previous examples, because it would make the small regions (inside the gray circles) very difficult to see. The impact of this elimination is very small in terms of objective quality, but the improvement in terms of subjective impact is quite large.

III. Performance Evaluation

To evaluate the proposed shape concealment technique, the Akiyo, Bream and Stefan video objects have been encoded according to the MPEG-4 Core Visual Object Type. For this, the following MPEG-4 video error resilience tools were used: resynchronization markers and data partitioning with reversible variable length codes. Additionally, since it was important (for

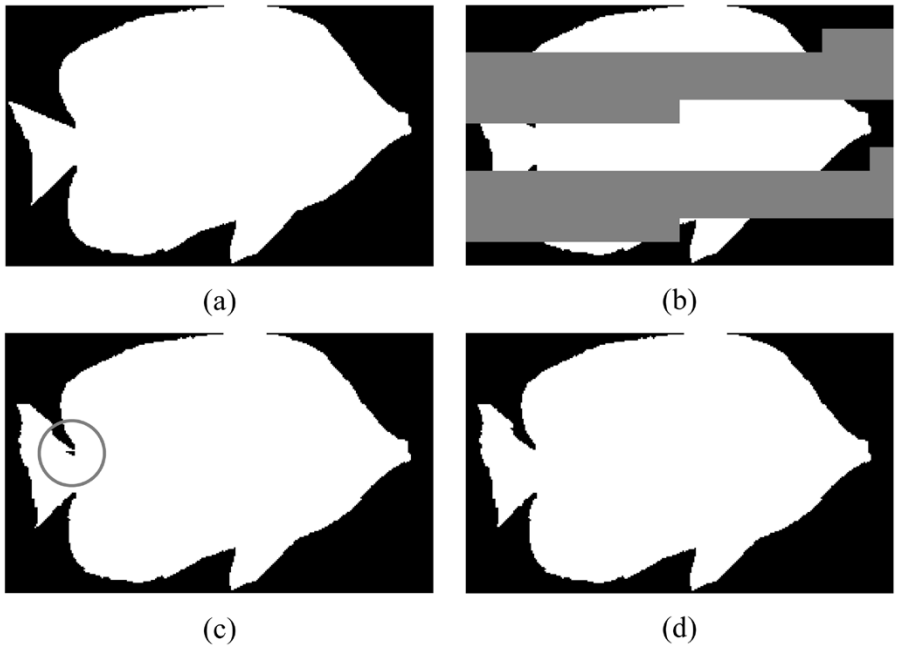

Fig. 12. Elimination of small transparent region outside but in contact with the concealed area. (a) Original uncorrupted alpha plane; (b) corrupted alpha plane; (c) concealed alpha plane before small regions elimination; (d) concealed alpha plane after small regions elimination.

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. The encoding parameters are summarized in Table I for each video object sequence.

To simulate an error-prone environment, instead of adding errors directly to the bitstreams, the decoder simply ignored the video packets randomly (following a uniform distribution which produces bursts of corrupted shape blocks). This allows the performance of the concealment technique to be evaluated independently of the decoder error detection capabilities. For each tested packet loss rate, each video object has been decoded 50 times, with 50 different error patterns or runs, while applying the proposed 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, due to the importance of the decoded texture quality for the estimation of the global motion parameters, numerical texture quality results will also be presented using the peak signal-to-noise ratio (PSNR) metric. However, the PSNR metric is only computed over the pixels that belong to both the concealed VOP being evaluated and the original VOP.

In the following numerical results, three different types of *Dn* and PSNR values are shown for the tested video object sequences: Dn_{low}, Dn_{avg}, and Dn_{high} and $\mathrm{PSNR}_{\mathrm{low}}$, $\mathrm{PSNR}_{\mathrm{avg}}$, and $\mathrm{PSNR}_{\mathrm{high}}$. While the Dn_{low} and Dn_{high} values correspond, respectively, to the *Dn* value associated with the best and the worst runs in terms of shape quality, Dn_{avg} corresponds to the average of the *Dn* values associated with the 50 different runs for each test case. As for $\mathrm{PSNR}_{\mathrm{low}}$, $\mathrm{PSNR}_{\mathrm{high}}$, and $\mathrm{PSNR}_{\mathrm{avg}}$, they have equivalent definitions. The *Dn* and PSNR values associated with a given run are simply the temporal average of the *Dn* and PSNR values, respectively, for all the VOPs in that video object sequence, defined as

$$Dn = \frac{1}{N_{\mathrm{VOP}}} \sum_{i=0}^{N_{\mathrm{VOP}}-1} Dn_i \tag{7}$$

$$\mathrm{PSNR} = \frac{1}{N_{\mathrm{VOP}}} \sum_{i=0}^{N_{\mathrm{VOP}}-1} \mathrm{PSNR}_i \tag{8}$$

where N_{VOP} is the total number of VOPs in the test sequence and i is the index corresponding to the VOP number. If, while decoding a bitstream, the decoder loses a complete VOP, the lost VOP is replaced with the previously decoded VOP, which is then used for the *Dn* and PSNR computations.

With the used encoding conditions, the error-free decoded texture qualities presented in Table II are obtained. The obtained error-free *Dn* is not included in the table since it is obviously 0.00% because the used shape coding (i.e., MPEG-4 shape coding) is lossless.

TABLE I
SUMMARY OF ENCODING PARAMETERS

	Akiyo	Bream	Stefan
Spatial Resolution	CIF	CIF	CIF
Frame Rate (fps)	10	10	15
Bit Rate (kbps)	64	128	128
VP Size (bits)	800	1600	1060

TABLE II
ERROR-FREE PSNR OBTAINED FOR THE CHOSEN ENCODING PARAMETERS

	Akiyo	Bream	Stefan
Error-free PSNR [dB]	33.85	30.01	30.46

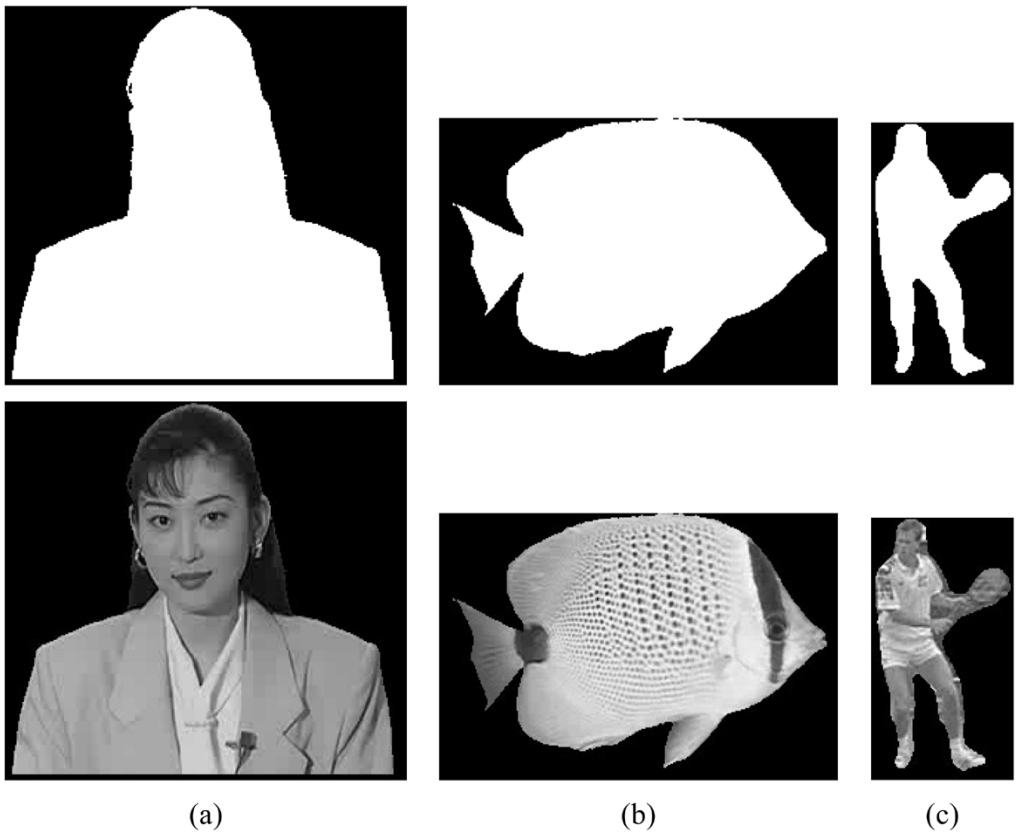

Fig. 13. Uncorrupted shape and texture of tested video objects. (a) Akiyo; (b) Bream; (c) Stefan.

TABLE III
DN VALUES FOR THE TESTED VIDEO OBJECT SEQUENCES FOR THE PROPOSED TECHNIQUE

Video packet loss rate	Dn [%] ($Dn_{low}/Dn_{avg}/Dn_{high}$)								
	Akiyo			Bream			Stefan		
1%	0.00	0.02	0.08	0.01	0.07	0.23	0.09	0.32	0.57
5%	0.03	0.08	0.23	0.19	0.34	0.62	0.91	1.66	2.55
10%	0.08	0.15	0.31	0.48	0.71	1.00	2.29	3.43	5.08
20%	0.20	0.29	0.42	1.15	1.49	2.18	6.03	7.41	9.75

TABLE IV
PSNR VALUES FOR THE TESTED VIDEO OBJECT SEQUENCES FOR THE PROPOSED TECHNIQUE

Video packet loss rate	PSNR [dB] ($PSNR_{low}/PSNR_{avg}/PSNR_{high}$)								
	Akiyo			Bream			Stefan		
1%	32.04	33.16	33.72	28.11	29.04	29.83	27.77	28.99	29.89
5%	29.64	30.74	31.57	25.04	26.49	27.51	23.81	25.04	26.20
10%	27.42	28.73	29.90	23.41	24.55	25.69	21.34	22.50	23.72
20%	25.36	26.14	27.55	21.04	22.02	23.08	18.95	19.61	20.37

As can be seen in Table III, even for relatively high video packet loss rates, the *Dn* values remain relatively low. In most cases, the *Dn* values are so low that they correspond to hardly noticeable artifacts (i.e., Dn_{avg} values around 1% and below). This is the case for the Akiyo video object with packet loss rates of 1%, 5%, 10%, and 20%, for the Bream video object with packet loss rates of 1%, 5%, and 10%, and for the Stefan video object with packet loss rates of 1%. For higher packet loss rates and less simple shapes, such as 20% for Bream and 5% for Stefan, the artifacts start to become more visible (i.e., Dn_{avg} values higher than 1% but still below 3%). And, finally, for the Stefan video object with a packet loss rate of 10%, the artifacts become clearly visible but still very acceptable (i.e., the Dn_{avg} value is 3.43%). For a packet loss rate of 20%, however, the Stefan artifacts become quite annoying (i.e., the Dn_{avg} value is 7.41%). These results clearly reflect the intrinsic shape concealment difficulty of the tested video objects, by ranking the Akiyo object as the easiest to conceal, followed by Bream and, finally, Stefan. This can be explained by considering the type of motion associated with the tested objects: the Akiyo object hardly moves, the Bream object moves quite a lot but suffers almost no deformation along time and, finally, the Stefan object moves a lot but also suffers a lot of deformation along time. This also helps explaining the fact that for the Akiyo video object less than 1% of the concealed shape blocks have to be refined because of local motion, while for the Bream video object 21% to 27%, depending on the video packet loss rate, have to be refined. As for the Stefan video object, which corresponds to the video object with the most local motion, 61% to 63% of the concealed shape blocks have to be refined, also depending on the video packet loss rate. In terms of texture quality, whose results are included in Table IV, the same comments that were made for the *Dn* values can be also made for the PSNR values.

In order to illustrate the results in Tables III and IV, the shape and texture of one decoded VOP is used for each video object sequence. For the Akiyo and Bream video objects, VOP 3 in the original 300 VOP sequence is used, as shown in Fig. 13(a) and

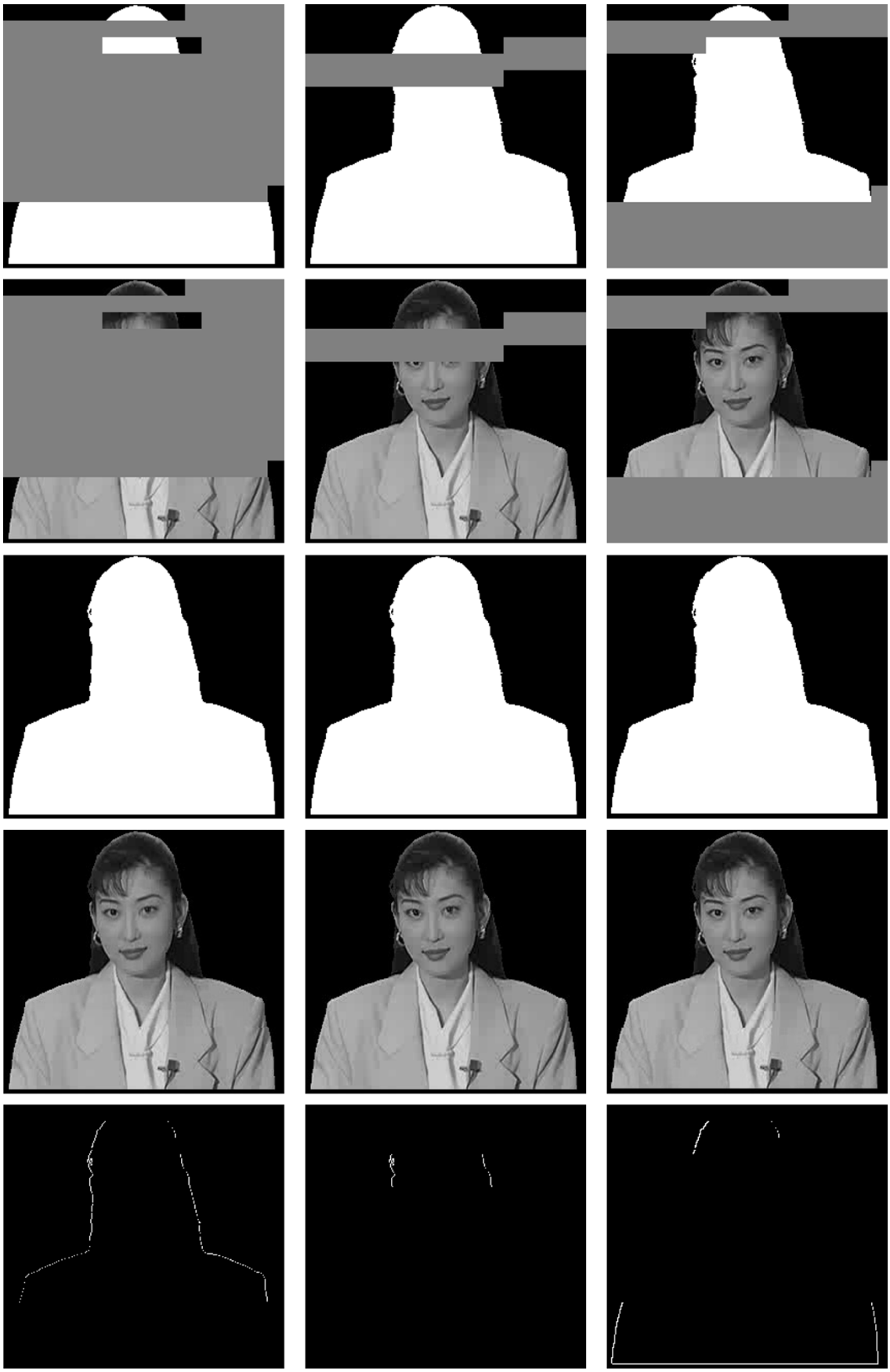

Fig. 14. Examples of corrupted and corresponding concealed alpha planes and textures, as well as difference images between concealed and original alpha planes, for the Akiyo video object with a video packet loss rate of 20%. (a) Error pattern 1; (b) error pattern 2; (c) error pattern 3.

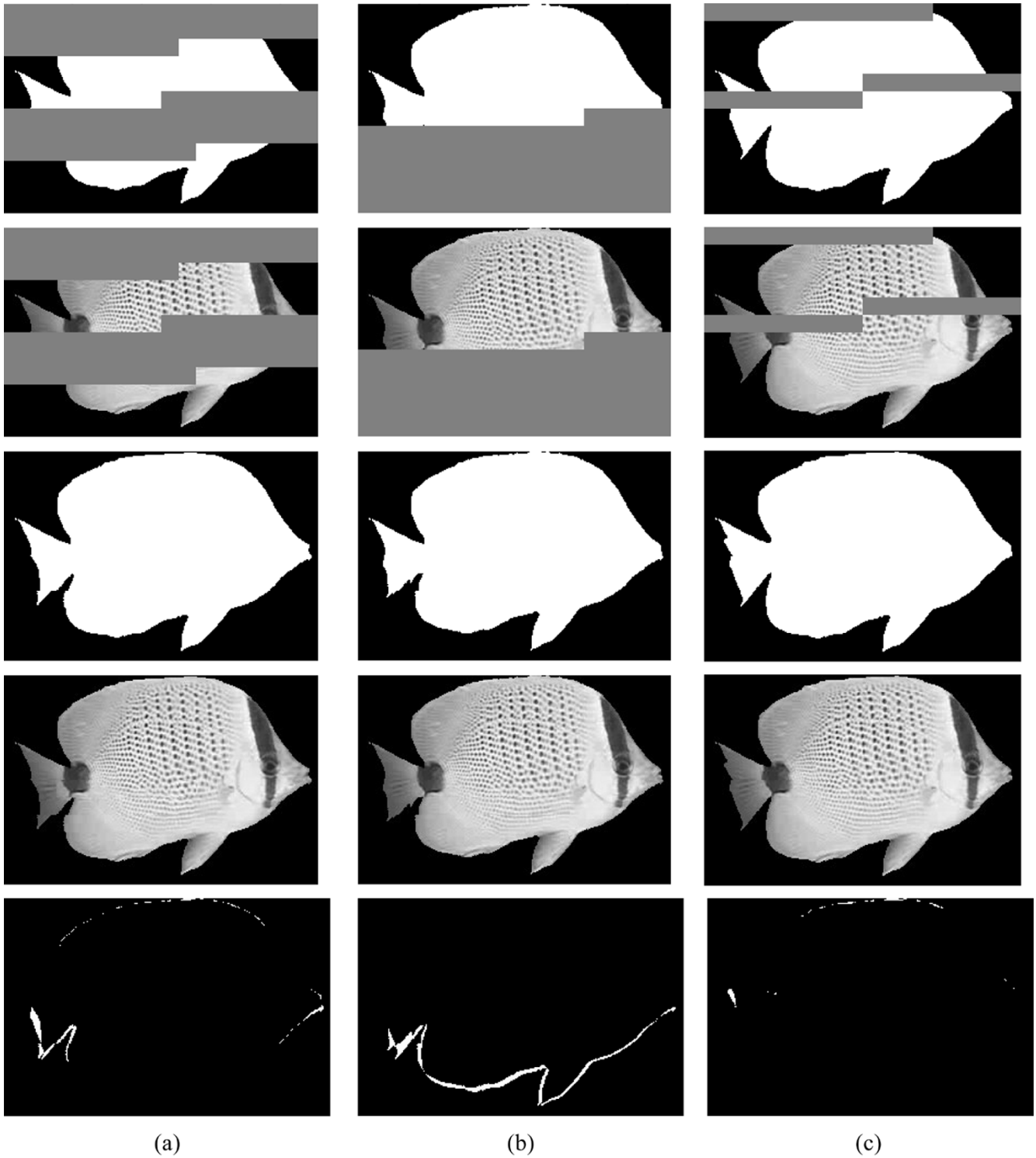

Fig. 15. Examples of corrupted and corresponding concealed alpha planes and textures, as well as difference images between concealed and original alpha planes, for the Bream video object with a video packet loss rate of 20%. (a) Error pattern 1; (b) error pattern 2; (c) error pattern 3.

(b), respectively. For the Stefan video object, VOP 2 in the original 300 VOP sequence is considered, as shown in Fig. 13(c). For each of the above VOPs, several examples of the corrupted alpha plane and texture are shown, in addition to the corresponding concealed alpha planes and textures, as well as difference images between the concealed and original alpha planes. Each example corresponds to one of the 50 different error patterns used above for a video packet loss rate of 20%. As shall be seen, these examples clearly reflect what was shown above by the numerical results.

For the Akiyo video object, as shown in Fig. 14, the obtained results are very good in the sense that the existing shape artifacts are unnoticeable. These concealment results were expected since the Akiyo object hardly moves and suffers no deformation along time. As for the *Dn* values corresponding to the shown concealed alpha planes, they are very low (typically

around or below 1%): 0.78% for pattern 1, 0.18% for pattern 2, and 1.15% for pattern 3. In terms of texture, the obtained results are also very good. As for the corresponding PSNR values, they are 28.24 dB for pattern 1, 32.85 dB for pattern 2 and 28.67 dB for pattern 3; the uncorrupted texture has a PSNR value of 35.24 dB. The large drop in the PSNR values is explained by the fact that large areas of the object are being concealed with global motion compensation. This means that the concealed texture can be slightly displaced (e.g., by one pixel) with respect to the original texture. This displacement goes unnoticed to the human user, leading to a very good subjective impact, but it can have a very large effect on the PSNR values, which are computed based on the differences between co-located pixels.

Next, in Fig. 15, the results for the Bream video object are shown. In terms of shape, the obtained results have a pleasing subjective impact on the viewer, in the sense that no annoying artifacts appear, except for error pattern 1, where the lower part of the back fin of the fish has been slightly displaced to the left. In fact, even for error pattern 2, which corresponds to the highest *Dn* value, the subjective impact is quite satisfactory. The *Dn* metric values corresponding to the shown concealed alpha planes are: 1.51% for error pattern 1, 3.49% for error pattern 2, and 0.46% for error pattern 3. In terms of texture, the same comments that were made for shape can also be made; the only annoying artifact that can be spotted is the texture associated with the lower part of the back fin, which has also been displaced with the shape. As for the corresponding PSNR values, they are 28.06 dB for pattern 1, 28.23 dB for pattern 2, and 29.53 dB for pattern 3; the uncorrupted texture has a PSNR value of 32.01 dB. The large drop in the PSNR values can be explained in the same way as for the Akiyo video object.

Finally, in Fig. 16, the results for the Stefan video object are shown. In this case, the shape artifacts are rather visible for error patterns 1 and 4, but the results are still tolerable for critical environments such as mobile networks. In terms of the *Dn* metric, the values are, as expected, higher than for the previous video objects because the motion associated with this video object is much more complex. The obtained *Dn* values corresponding to the shown concealed alpha planes are: 11.50% for error pattern 1, 2.26% for error pattern 2, 8.86% for error pattern 3, and 7.58% for error pattern 4. In terms of texture, the artifacts are also rather visible, which is easily understandable since they are associated with shape displacements (in the head for error pattern 2 and in the racket for error patterns 1 and 4). However, as said above, they may be rather tolerable for critical environments, although it will depend on the specific application. As for the corresponding PSNR values, they are 24.41 dB for pattern 1, 26.64 dB for pattern 2, 26.91 dB for pattern 3, and 21.91 dB for error pattern 4; the uncorrupted texture has a PSNR value of 30.31 dB. The explanation for the large drop in the PSNR values is basically the same as for the other two video objects, but here in addition to the global motion compensation a lot of local motion refinement has also been applied.

In addition to these results, the technique proposed here can also be compared to other temporal shape error concealment techniques, such as those proposed in [8] and [9]. Of these two techniques, the one that represents the most direct competition

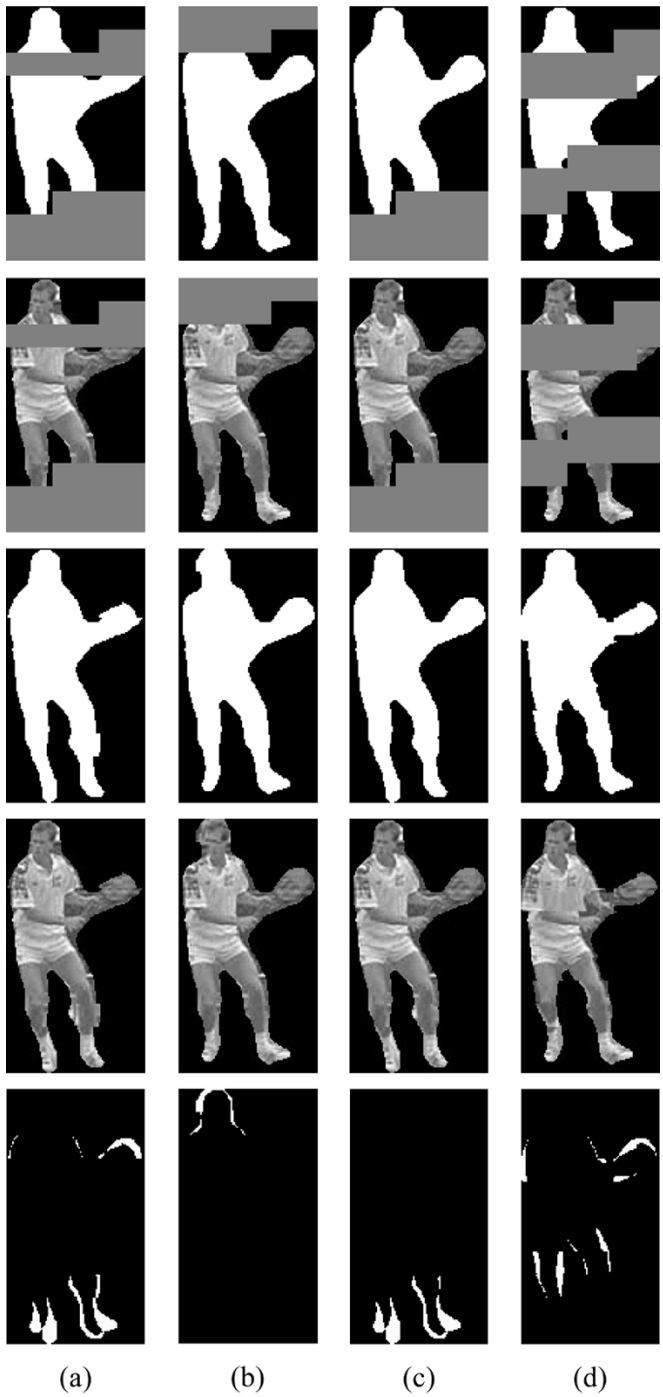

Fig. 16. Examples of corrupted and corresponding concealed alpha planes and textures, as well as difference images between concealed and original alpha planes, for the Stefan video object with a video packet loss rate of 20%. (a) Error pattern 1; (b) error pattern 2; (c) error pattern 3; (d) error pattern 4.

TABLE V
DN VALUES FOR THE TESTED VIDEO OBJECT SEQUENCES FOR THE AMV TECHNIQUE

Video packet loss rate	Dn [%] (Dn_{low}/Dn_{avg}/Dn_{high})								
	Akiyo			Bream			Stefan		
1%	0.00	0.00	0.02	0.02	0.10	0.33	0.16	0.50	0.90
5%	0.01	0.02	0.03	0.39	0.57	0.85	1.70	2.58	3.60
10%	0.02	0.04	0.06	0.85	1.19	1.64	4.05	5.23	6.54
20%	0.06	0.09	0.13	2.09	2.50	3.34	9.24	10.89	14.06

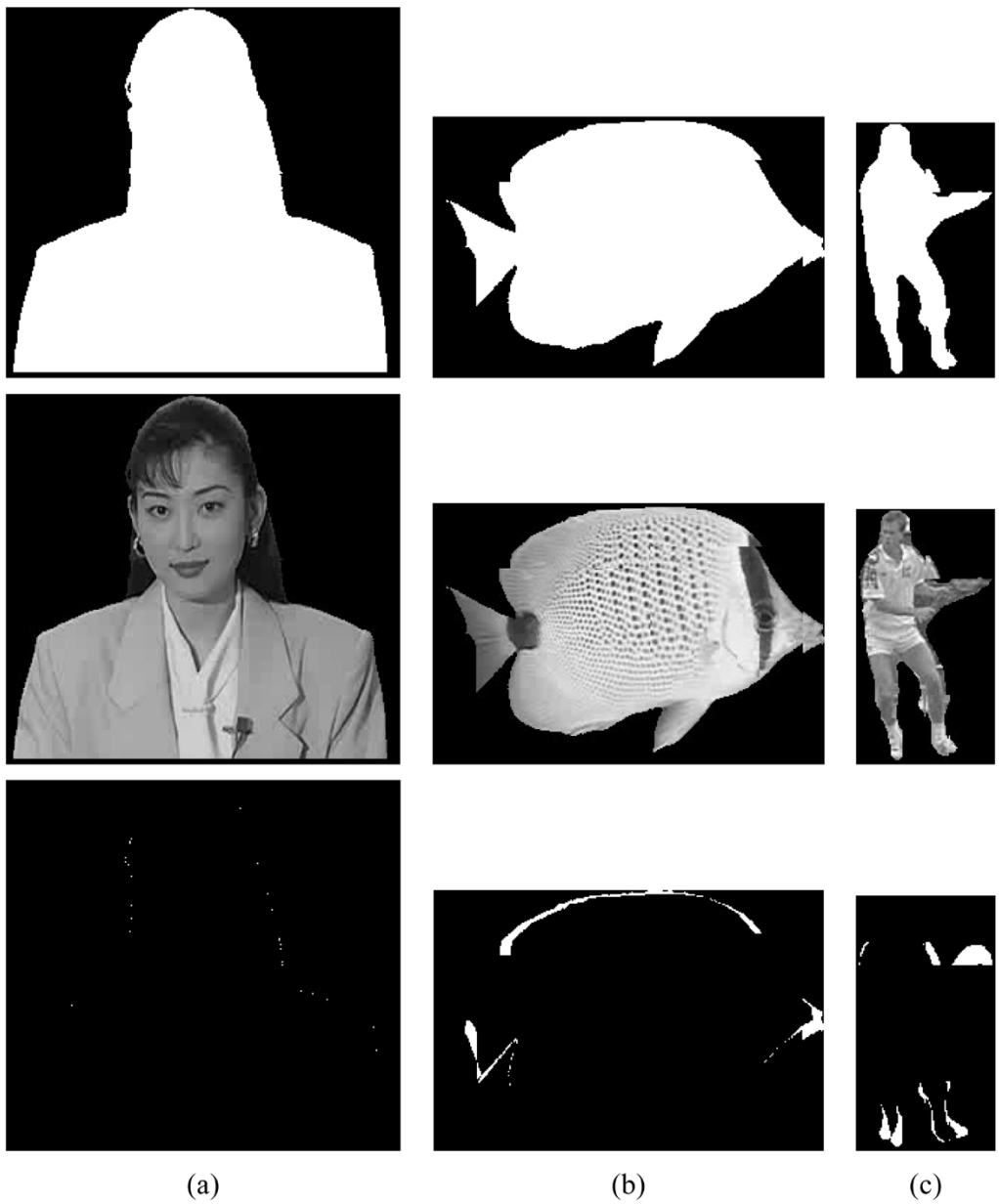

Fig. 17. Concealed shape and texture of tested video objects with the AMV technique, as well as difference images between concealed and original alpha planes. (a) Akiyo; (b) Bream; (c) Stefan.

to the technique proposed here is clearly the one proposed in [8] since the one proposed in [9] involves the use of additional information sent by the encoder. Therefore, the one proposed in [8] was chosen for a direct comparison. As explained in Section I, three different approaches are used in [8] to conceal corrupted shape blocks based on the motion compensation of information from the previous time instant. Of the three approaches, the one that leads to the best objective shape quality will be considered here; this approach uses the motion vector associated with the block located above the corrupted block, in the current time instant, to find a replacement shape block in the previous time instant [referred to in the following as above motion vector (AMV) technique].

Based on this decision, all the corrupted bitstreams that were used before to obtain the results shown in Tables III and IV have been decoded again, but this time concealment was done with the AMV technique proposed in [8]. The obtained *Dn* values are shown in Table V, which is arranged in the same manner as Table III in order to make the comparison easier. As can be seen, when the AMV technique is used, the *Dn* values are slightly lower for the Akiyo video object, which corresponds to slightly better quality than what was achieved with concealment technique proposed in this paper. This difference, however, is so small and already in such a high quality range that it cannot be considered significant and, therefore, it can be said that in this case both concealment techniques have the same performance. For the other video objects, however, the *Dn* values obtained with the AMV technique are significantly larger than those obtained for the technique proposed here, corresponding to visually more noticeable artifacts. Based on these results, it can be said that the technique proposed in this paper is always equal to or better than the one proposed in [8].

As for the PSNR values obtained with the AMV technique, these are not shown here because the AMV technique is not influenced by the texture quality as the one proposed in this paper is (i.e., the texture is used to estimate the global motion parameters).

In order to illustrate the results in Table V, some of the previously used corrupted alpha planes and textures will be reused here; all the corrupted alpha planes and textures, corresponding to error pattern 1 in the previous examples, have been concealed with the AMV technique and are shown in Fig. 17. As can be seen by comparing the concealed alpha planes and textures shown in Fig. 17 with the corresponding alpha planes and textures concealed with the technique proposed in this paper, the performance is only similar for the Akiyo video object. However, for both Bream and Stefan, the performance of the shape concealment technique proposed here is much superior, with much less visible artifacts, which allows to conclude that the proposed technique works better when object motion is more intense.

Even though a direct comparison of the technique proposed here and the one proposed in [9] was not performed, some comments can still be made. For instance, for very simple cases where the shape motion can be described in terms of global motion alone, it can be expected that the technique proposed here be slightly inferior to the one proposed in [9], where global motion parameters are computed at the encoder using uncorrupted video objects. However, the technique proposed here eliminates the very limiting disadvantage of having the decoder rely on specific data produced by the encoder, which is a rather unrealistic scenario at least in the context of applications using coding standards (in fact it would only work for encoder-decoder combinations from the same manufacturer unless the auxiliary stream with global motion data would be standardized). For other more realistic cases, where the shape motion can not be described in terms of global motion alone, the technique proposed here will clearly outperform the one proposed in [9]. The disastrous effects of trying to conceal this type of shape data with global motion compensation alone were illustrated in Fig. 3.

IV. Final Remarks

In this paper, a temporal shape concealment technique was proposed to conceal shape errors in binary alpha planes or in the binary supports of gray scale alpha shapes corresponding to the objects in object-based video coding systems, such as the MPEG-4 standard. This technique is based on a combination of global motion and local motion compensation. First, it starts by assuming that the alpha plane changes occurring in consecutive time instants can be described by a global motion model and simply tries to conceal the corrupted alpha plane blocks by using the corresponding blocks in the previous global motion compensated alpha plane. However, since not all alpha plane changes can be perfectly described by global motion, an additional local motion refinement is then applied to deal with areas of the object that have significant local motion. Results have been presented showing that this technique is able to recover lost shape data with rather small distortion, even for cases where local motion exists and the global motion model is not able to perfectly describe the shape motion.

Finally, to end this paper, 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 achieved is highly dependent on the quality of the shape data. Additionally, it is also important not to forget that many of the object-based functionalities depend on the quality of the shape data. Therefore, the role of robust shape error concealment techniques in making object-based video applications possible in error-prone environments will be very important.

References

[1] *Information Technology—Coding of Audio-Visual Objects*, ISO/IEC 14 496-2, Dec. 1999.

[2] S. Shirani, B. Erol, and F. Kossentini, "A concealment method for shape information in MPEG-4 coded video sequences," *IEEE Trans. Multimedia*, vol. 2, no. 3, pp. 185–190, Sep. 2000.

[3] X. Li, A. K. Katsaggelos, and G. M. Schuster, "A recursive shape error concealment algorithm," in *Proc. Int. Conf. Image Processing*, vol. 1, Rochester, NY, Sep. 2002, pp. 177–180.

[4] X. Li, G. M. Schuster, and A. K. Katsaggelos, "Curve-fitting algorithms for shape error concealment," in *Proc. Nordic Signal Processing Symp.*, Hurtigruten, Norway, Oct. 2002, pp. 4–7.

[5] G. M. Schuster, X. Li, and A. K. Katsaggelos, "Spline based boundary loss concealment," in *Proc. Int. Conf. Image Processing*, Barcelona, Spain, Sep. 2003.

[6] L. D. Soares and F. Pereira, "Spatial shape error concealment for object-based image and video coding," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 586–599, Apr. 2004.

[7] G. M. Schuster, X. Li, and A. K. Katsaggelos, "Shape error concealment using hermite splines," *IEEE Trans. Image Process.*, vol. 13, no. 6, pp. 808–820, Jun. 2004.

[8] M. R. Frater, W. S. Lee, M. Pickering, and J. F. Arnold, "Error concealment of arbitrarily shaped video objects," in *Proc. Int. Conf. Image Processing*, vol. 3, Chicago, IL, Oct. 1998, pp. 507–511.

[9] P. Salama and C. Huang, "Error concealment for shape coding," in *Proc. Int. Conf. Image Processing*, vol. 2, Rochester, NY, Sep. 2002, pp. 701–704.

[10] J. Zhang, J. F. Arnold, M. R. Frater, and M. R. Pickering, "Video error concealment using decoder motion vector estimation," in *Proc. IEEE Region 10 Conf. (TENCON)*, vol. 2, Brisbane, Australia, Dec. 1997, pp. 777–780.

[11] A. Zakhor and F. Lari, "Edge based 3-D camera motion estimation with application to video coding," *IEEE Trans. Image Process.*, vol. 2, no. 4, pp. 481–498, Oct. 1993.

[12] M. Bierling and R. Thoma, "Motion compensating field interpolation using a hierarchical structure displacement estimator," *Signal Process.*, vol. 11, no. 4, pp. 387–404, Dec. 1988.

[13] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C—The Art of Scientific Computing*, 2nd ed. Cambridge, U.K.: Cambridge Univ. Press, 1992.

Luis Ducla Soares (S'98–M'04) was born in Lisbon, Portugal, in March 1973. He received the *Licenciatura* and Ph.D. degrees in electrical and computer engineering from Instituto Superior Técnico (IST), Universidade Técnica de Lisboa, Lisboa, Portugal, in 1996 and 2004, respectively.

He is currently an Assistant Professor at the Information Science and Technology Department of Instituto Superior de Ciências do Trabalho e da Empresa (ISCTE), Portugal, and a member of the research staff of the Telecommunications Institute, Portugal. His research interests include object-based error resilience as well as video coding.

Fernando Pereira (S'92–M'93–SM'99) was born in Vermelha, Portugal, in October 1962. He received the *Licenciatura*, M.Sc., and Ph.D. degrees in electrical and computer engineering from Instituto Superior Técnico (IST), Universidade Técnica de Lisboa, Lisboa, Portugal, in 1985, 1988, and 1991, respectively.

He has been participating in the work of ISO/MPEG for many years, notably as the head of the Portuguese delegation, chairman of the MPEG Requirements group, and chairing many ad hoc groups related to the MPEG-4 and MPEG-7 standards. His current areas of interest are video analysis, processing, coding and description, and multimedia interactive services. He is a member of the Scientific and Program Committees of tens of international conferences and workshops. He has contributed more than 150 papers to journals and international conferences.

Dr. Pereira is currently a Professor with the Electrical and Computers Engineering Department at IST. He is responsible for the participation of IST in many national and international research projects. He is a member of the Editorial Board and Area Editor on Image/Video Compression of the *Signal Processing: Image Communication Journal* and an Associate Editor of IEEE TRANSACTIONS OF CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, IEEE TRANSACTIONS ON IMAGE PROCESSING, and IEEE TRANSACTIONS ON MULTIMEDIA. He won the 1990 Portuguese IBM Award and an ISO Award for Outstanding Technical Contribution for his participation in the development of the MPEG-4 Visual standard, in October 1998.