SHAPE CODING IN MPEG-4: AN OVERVIEW

Signal Compression, ECE 242, Project Report

Sandeep Bhat

sandeepkbhat@ece.ucsb.edu

Contents

1. Introduction

Applications like content-based storage and retrieval, studio and TV postproduction applications requiring editing of video content, mobile multimedia applications requiring content-based interactivity and security applications requiring content-based scalability all share a common requirement: video data should be accessible on a object basis. This is the primary motivation for development of shape coding of video data based on object descriptions.

Conventional video coding describes the video objects (VO) only based on texture (luminance, chrominance) and motion. For the above said applications video objects need to be described by shape also. In addition to the object-based video representation, the use of shape was aimed at obtaining better subjective picture quality and improved coding efficiency.

MPEG-4 Visual provides all these functionality by allowing arbitrarily shaped video objects to be transmitted. Every frame of the VO, also called the video object plane (VOP), consists of shape and texture information along with the optional motion information. Figure 1.1 shows the decomposition of video frame into VOs in MPEG-4 Visual. Several VOs are transmitted along with the composition information so that the decoder can put them together into a video scene.

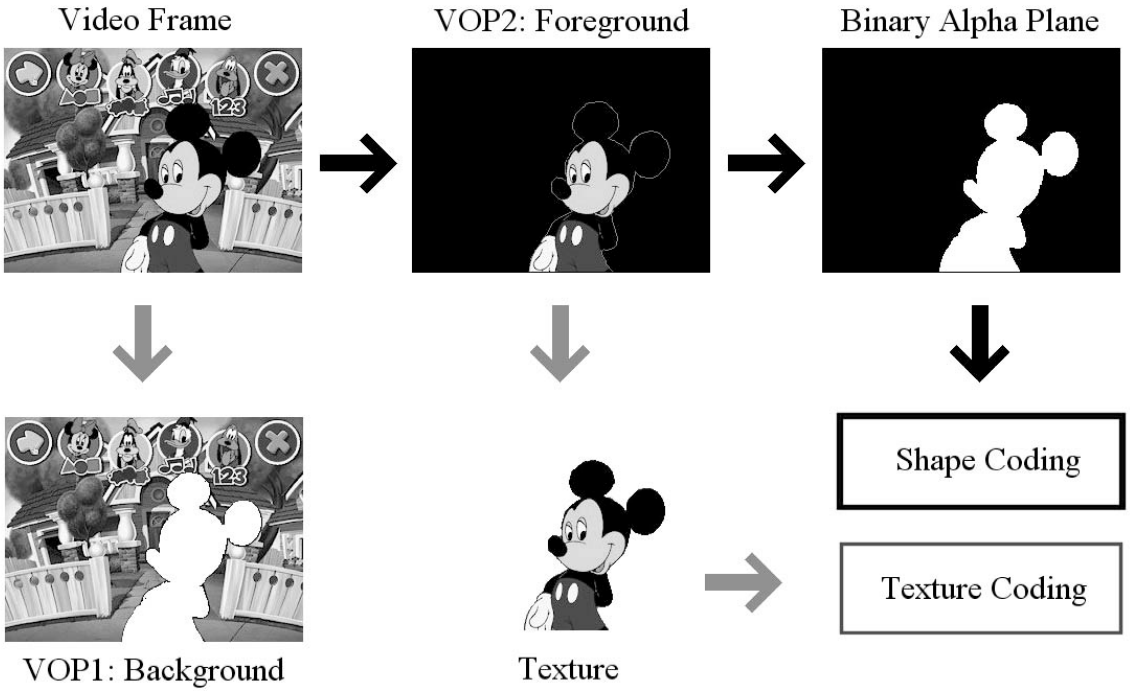

Figure 1.1: Decomposition of video frame in MPEG-4. Source [1]

This report is organized as follows. Section 2 covers the basics of shape coding. Section 3 covers the various shape-coding tools available in MPEG-4. In section 4 several algorithms that have been developed to model R-D characteristics are discussed and various rate control techniques are presented. In sections 5 and 6 various post-processing schemes are discussed. Finally in section 7 several applications that can benefit by using shape based coding have been presented.

2. Shape Coding

2.1 Overview

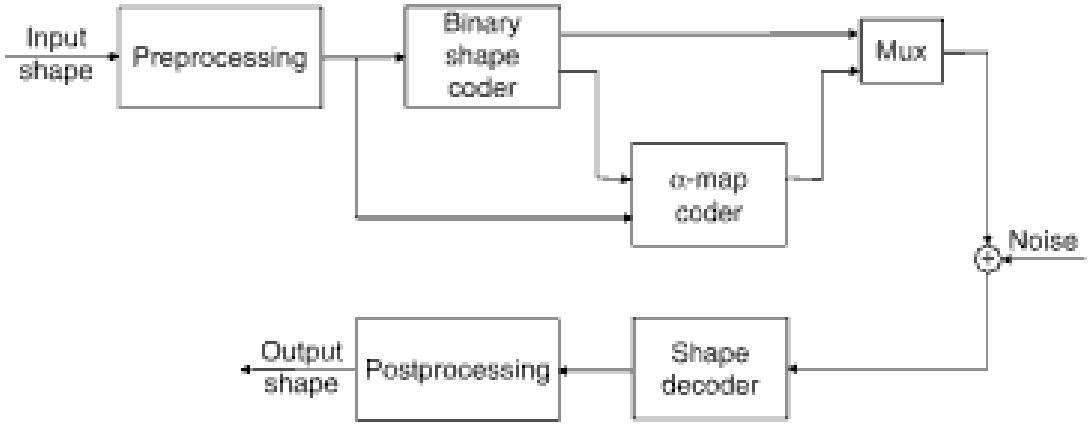

Figure 2.1: Processing steps for shape coding considering binary and gray-scale α-maps. Source [2]

Figure 2.1 illustrates the steps in shape coding. The optional preprocessing step can remove noise and increase coding efficiency by simplifying the shape to be coded. The preprocessed shape information is separated into binary shape defining the pels that belong to the shape and gray-scale information defining the transparency of pels in the object. The α–map coder then codes these information. This coded data is then multiplexed with the composition information and transmitted to the decoder. The decoding is done by the shape decoder and optional post processing step provides boundary smoothing and error concealment. The decoded VOs are then composed into a scene using the composition information.

2.2 Classes of shape coders

There are three classes of binary shape coders. Each of them is described in the following sections.

2.1.1 Implicit Shape Coding

This is also referred to as chroma keying. In this class of shape coding, the background is defined by either a specific pixel value or a class of pixel values. The object is represented using the remaining pels. One of the pixel values can be used to define transparent pels (that are not displayed by the receiver). GIF89a uses an implicit shape coder to represent arbitrarily shaped objects. The major disadvantage of implicit shape coding is that it needs good texture coding to attain good shape coding. As humans are more sensitive to shape distortions than texture distortions, this coupling is undesirable for many applications that need shape coding.

2.1.2 Bitmap-Based Shape Coding

This class of coder encodes if the pel belongs to the object or not. The modified read (MR) code used in fax G4 is a type of bitmap-based coder. MR encodes the 'reference' lines using modified Huffman code (MH). The next line is compared to the reference line, the differences determined, and then the differences are encoded and transmitted. This is effective as most lines differ little from their predecessor.

2.1.3 Contour-Based Shape Coding

Several types exist in this class of shape coders. For lossy and lossless encoding of object boundaries chain coders and polygon approximations are used. Certain techniques like Fourier descriptors that were originally developed for recognition purposes have also been applied to shape coding.

Following the contour of an object, the chain coder encodes the direction in which the next boundary pel is located. Different variations of this algorithm exist based on the number of neighboring pels inspected and the type of grids (rectangular, hexagonal) used. The Figure 2.2 shows the chain coder patterns for a rectangular grid of 4 and 8 neighboring pels. This technique however is not very effective for inter-mode (where temporal prediction is used).

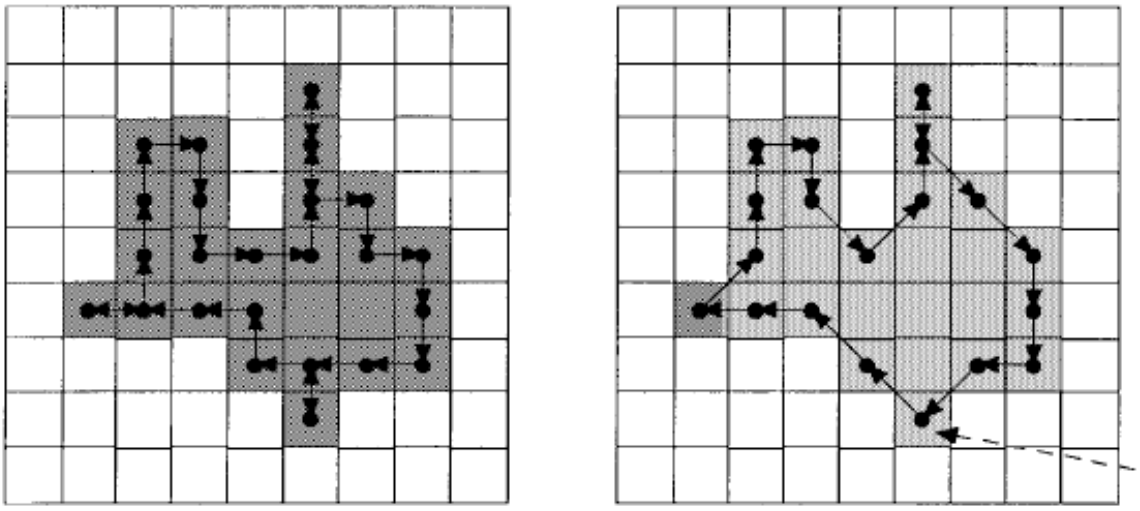

Figure 2.2 A chain code follows the contour of an object by describing the direction from one boundary pel to the next. The dotted line in the right figure shows the starting point. Source [3].

Polygon-based shape representations can be used in inter and intra mode. The method tries to approximate the contour by using polygons. The approximations are refined using the maximum Euclidean distance between the approximated and the original contour (d_{max}) as a quality measure. For temporal prediction, the texture motion vectors are

applied to the vertices of the previously coded shape defining the predicted shape for the current shape. Then, all vertices within the allowable approximation error d_{max} define the new polygon approximation. Figure 2.3 shows the polygon approximation iteration.

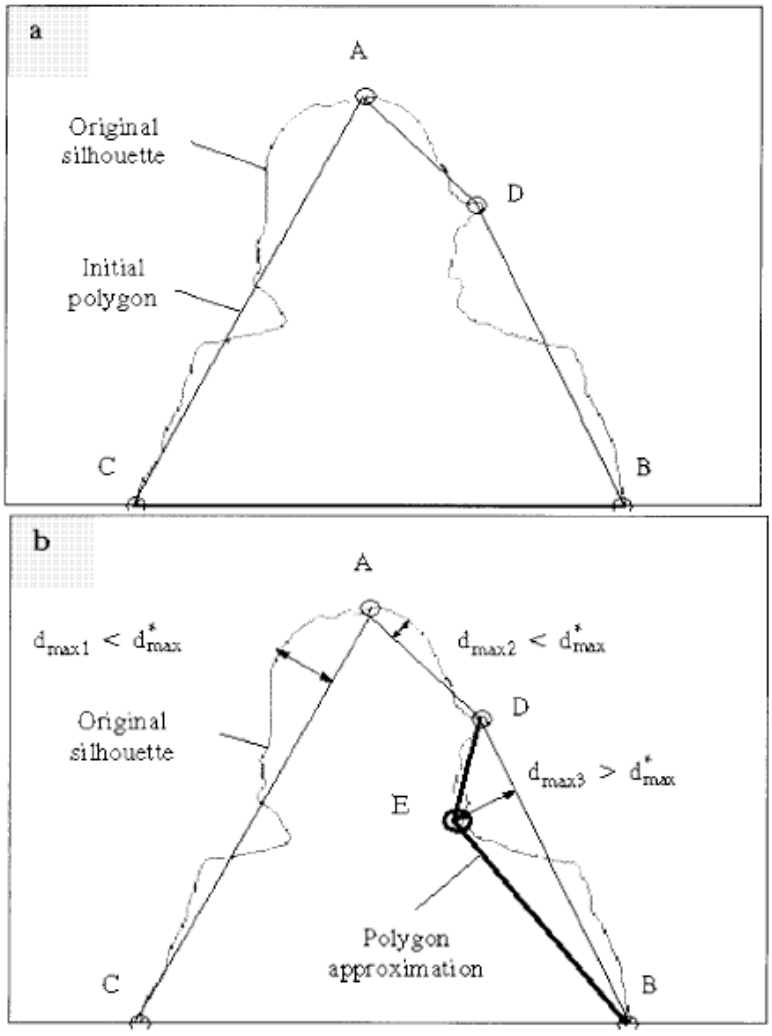

Figure 2.3: Polygon-based shape approximation. Source [3].

3. MPEG-4 Shape Coding Tools

MPEG-4 uses the binary context adaptive arithmetic coder for shape coding. This is discussed in detail in this section.

3.1 Representation of 2D shape

Shape is defined by an α–map M_k of size $X*Y$ pels, where X and Y are the width and height of the VOP.

$$M_k = \{m_k(x,y \mid 0 \le x \le X, 0 \le y \le Y\}, 0 \le m_k \le 255$$

The shape M_k defines if the pel belongs to the object ($m_k(x,y) > 0$) or not ($m_k(x,y) = 0$). α-values of 255 are used for opaque objects and α-values ranging from 1 to 255 are used for transparent objects. The spatial and temporal resolution of the α-map remains the same as that of the video luminance signal. Let $\mathbf{x} = (x,y)^T$, $s_b(\mathbf{x})$ be the background image, $s_o(\mathbf{x})$ be the object and $M_o(\mathbf{x})$ be the corresponding α-map. Then the following equation shows how the object is overlaid on the background.

$$s(\mathbf{x}) = \left(1 - \frac{M_o(\mathbf{x})}{255}\right) s_b(\mathbf{x}) + \frac{M_o(\mathbf{x})}{255} s_o(\mathbf{x})$$

Figure 3.1 shows how the amplitude of the α-map decides the visibility of the object in the scene.

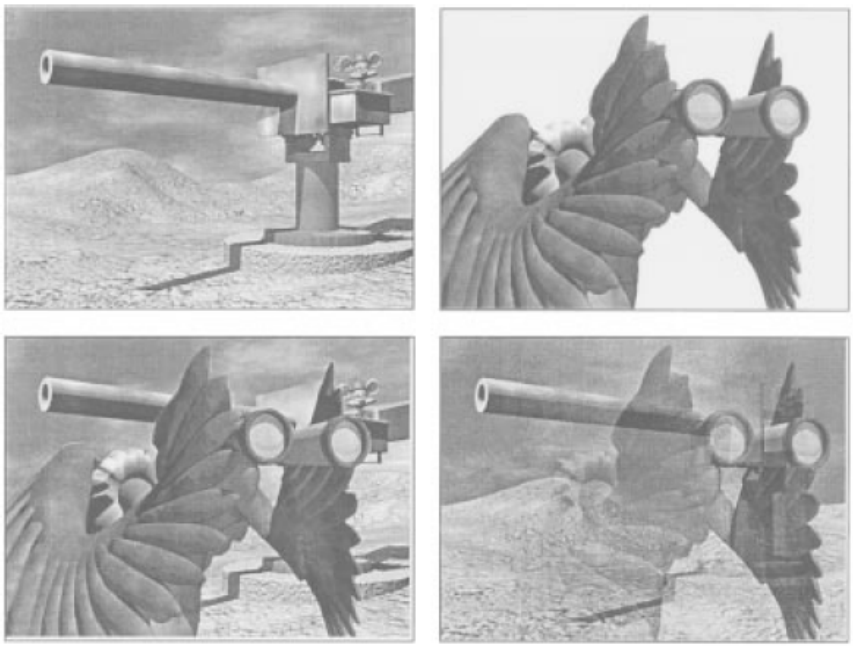

Figure 3.1: Composing the objects in a scene using the α-map. The top-left image is the background and the top-right image is the object. The bottom-left image is composed with constant transparency and bottom-right image with gray-scale transparency. Source [3].

3.2 Binary Shape Coding

The binary shape coder used in MPEG-4 is called the context-based arithmetic encoder (CAE). It works on macroblocks (MB) of 16 x 16 pixels that are defined for every VO in MPEG-4. The MBs can be of the transparent, opaque or boundary type. The alpha values for each MB are coded separately. The coding itself comprises of arithmetic coding of the alpha values depending on the context that has been computed for each pel. MPEG-4 has defined a table of statistical probabilities [5] for every possible context. For coding a given pel, the statistical probability is chosen based on its context and codewords are then assigned based on the probability.

Computation of the context depends on the mode of encoding (the intra-mode or the inter-mode). Figure 3.2 shows the templates that are used to compute the context of a pel for the intra and inter modes. The MBs are referred to as binary alpha blocks (BABs).

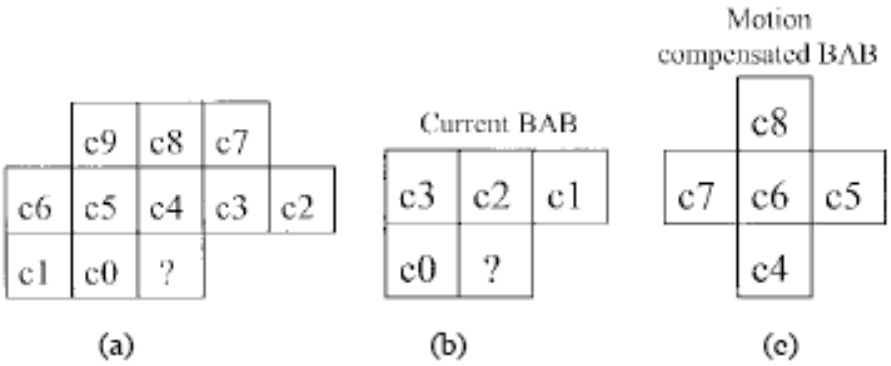

Figure 3.2: (a) This template shows the arrangement of the context bits for intra CAE. (b) The two parts of the inter template are shown, with bits c0–c3 coming from the current BAB and spatially related to the pixel to be decoded "?" as indicated, and bits c4–c8 coming from the motion-compensated BAB with c6 fully aligned with "?." Source [4,7].

$c_k = 0$ for transparent pels and $c_k = 1$ for opaque pels. As seen in the figure in the inter-mode the temporal redundancy is exploited by using pels from the corresponding motion compensated BAB in the reference frame (Figure 3.2c) To avoid ambiguity the following rules are used when building contexts:

- Pels of the context that are outside the VOP are taken to be zero (i.e. assumed transparent).
- If there are pels from BABs to the right of the current BAB, which are yet to be coded (i.e. their values are unknown), their values are taken to be the same as their bordering pels (closest neighbors in the MB).

The context is then described by a bit pattern of 10 and 9 bits for the intra and inter-modes, respectively and it is given by:

$$C = \sum_k c_k \cdot 2^k$$

where c_k is the binary alpha values corresponding to the positions described in the templates of Figure 3.2. Using C, the probability of current pel to be 0 or 1 is then estimated by table look-up. This is then used for arithmetic coding.

To increase coding efficiency, MPEG-4 allows lossy coding. This is done by subsampling the MB by a factor of 2 or 4 (into subblocks of size 8 x 8 pixels or 4 x 4 pixels). The inverse of these factors is sometimes referred to as the conversion ratio (CR). The subblocks are then encoded as described earlier by the CAE and transmitted along with the CR. The decoder decodes the subblocks and up-samples them to the original MB size. The decoded shape might not be the same as the original. To remove blockiness in the decoded shape, MPEG-4 suggests the use of an adaptive non-linear up-sampling filter as shown in Figure 3.3. The up-sampled pel is filled using the weighted sum of the pels in its context.

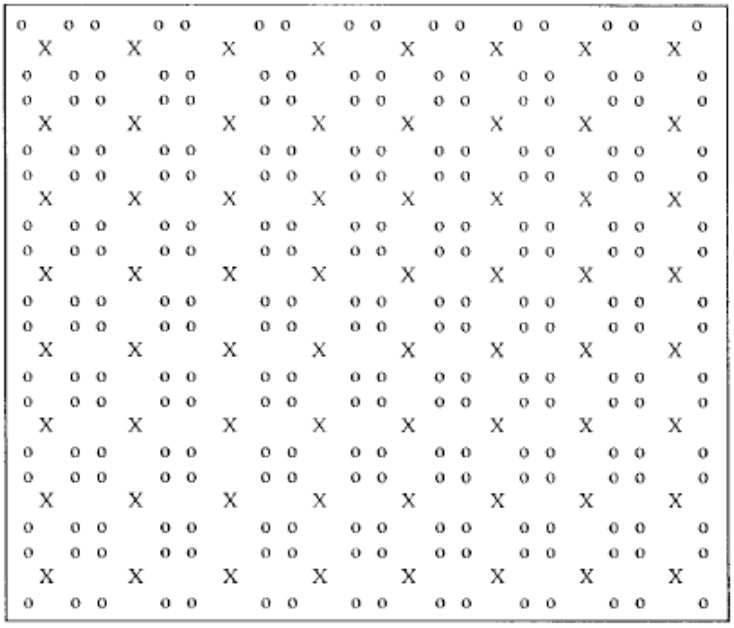

Figure 3.3: The spatial relationships between the subsampled pixels of a reconstructed 8 x 8 pixel BAB denoted by 'X' and the up-sampled pixels constituting the 16 x 16 BAB denoted by 'o.'. Source [4]

The shape coder efficiency depends on the orientation of the shape data. Hence the MBs are sometimes transposed by the encoder prior to coding.

In the inter-mode the shape motion vectors (MV) are found for the shape, similar to MVs for texture. These are predicted from previous coded VOPs. In the absence of shape MVs the texture MVs are used for prediction. These motion vectors are then used to align the context templates as shown in Figure 3.2.

3.3 Criteria for Evaluating Coding Efficiency

MPEG-4 describes two criteria for measuring the quality of coded shape objectively. One is the maximum of the minimum Euclidean distance d^*_{max} between each coded contour point and the corresponding point on the original contour. If breaks are introduced in the contour due to lossy coding, this does not serve as a good measure. In such cases a second measure d_n is used. It is defined as the ratio of the wrongly represented pixels in the coded shape to the total number of pixels in the original shape. It is easy to see that this measure is shape dependent. The evaluation of the coding based on these objective measures has shown that they actually do reflect the subjective quality of the video.

3.4 Gray-Scale Shape Coding

Gray-scale α-maps are represented by 8-bit α-values, with each value defining the pel transparency. α-maps of the binary type are coded using the binary shape coder described previously. For those of the arbitrary type, the outline of the object is first encoded as binary shape and the actual α-map is coded using the texture coding tools of MPEG-4 (DCT, padding, motion compensation).

3.5 Coding of Boundary Macroblocks

For arbitrarily shaped VO, the reference VOP used for prediction of motion vectors (MV) is also of arbitrary shape. This could lead to inefficient prediction if the MV refers to transparent pels in the reference VOP. Figure 3.4 illustrates three such situations.

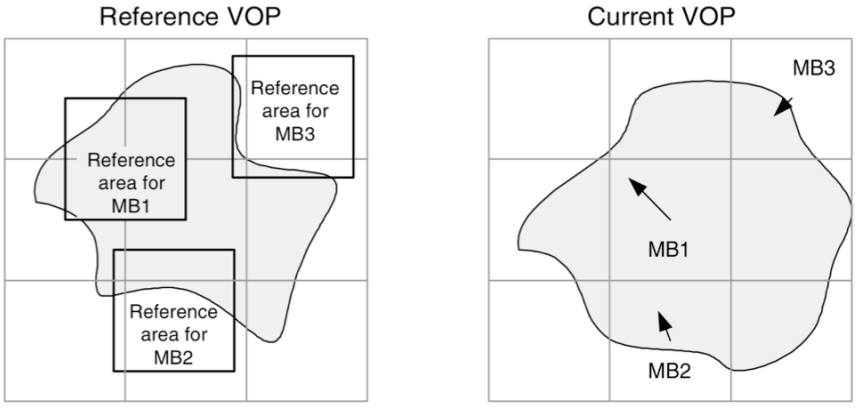

Figure 3.4: MB1 is entirely opaque but its MV points to region in the reference VOP that contains transparent pixels. Source [6]

This is avoided by predicting the MVs using a padded version of the previously decoded VOP. The 'transparent' pixels in each boundary MB of a reference VOP are extrapolated horizontally and vertically from opaque pixels as shown in Figure 3.5.

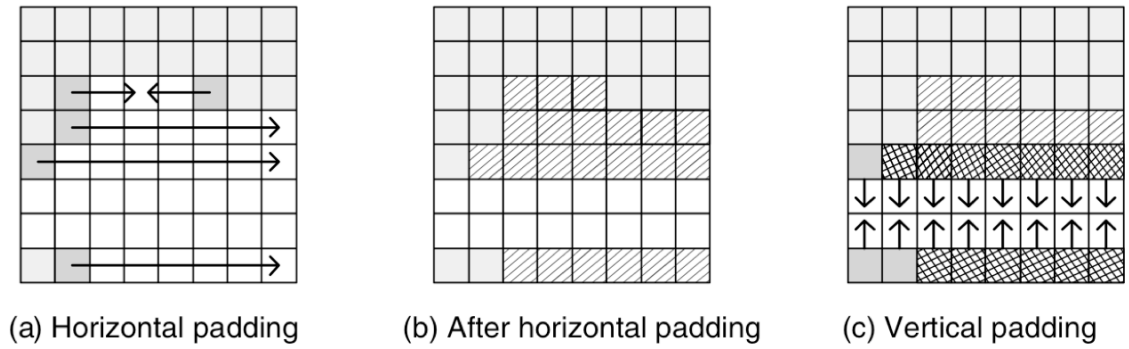

Figure 3.5: Horizontal and vertical padding in boundary MB. Source [6]

The extrapolation is done according to the following rules:

- Opaque pels at the edge of the BAB (dark grey cells in Figure 3.5a) are extrapolated horizontally to fill the transparent pels in the same row. The value is simply copied if a row is bordered by opaque pels only on one side. If the row has two bordering opaque pels then their average value is used to fill the transparent pels.
- After horizontal extrapolation is complete, the same procedure is repeated in the vertical direction. The transparent pels filled in the previous step are no longer considered transparent.

4. Bit-rate Handling

4.1 Rate-Distortion Models

Several R-D models exist for texture coding, that help in choosing a quantizer value needed to meet the specified rate constraints. These models can also be used to analyze different sources so as to optimize the coding efficiency in a computationally efficient manner. Analogous models have been developed for handling rate constraints for shape coding. The main motivation for these models is to be able to provide accurate R-D characteristics without performing the actual coding operation. In MPEG-4 shape coding, down–sampling and arithmetic coding are done to compute the rate, and up-sampling and difference calculations are done to compute the distortion at the various scales. Accurate R-D estimates can help in optimal allocation of bits for the binary shape among the various MBs and also lower the complexity of bit allocation between shape and texture coding.

The problem of modeling the rate can be formalized as follows. Consider the R-D values for a binary shape that is coded at resolution *k*. If *f(·)* denotes the modeling function and θ_i denotes the features extracted from the shape, we can pose the modeling problem as

$$(R,D)_k \approx f(\theta_i)$$

Several approaches have been proposed to solve this problem. One approach called state partitioning [8] attempts to solve this by using all possible binary patterns over 2 x 2 pel neighborhood and categorizing them into N states. In the training mode, the rate corresponding to each of these states is calculated. The average rate for a given shape can then be found by just counting the occurrences of different states. This scheme is computationally efficient. However it doesn't model the distortion accurately as distortion depends on a number of neighborhood pels and estimating them using 2 x 2 pel neighborhoods is insufficient.

To overcome the drawback of this above approach a Markov random field (MRF) based model has been proposed [9]. Edge, line and noise, the three parameters needed for the model, are estimated from the VOP using the histogram method. These are then used to calculate the statistical moments of the MRF model (particularly the Chein model) which are then used as inputs to a simple feed-forward network. The output of the network is the estimated rate and distortion of shape data at various scales. The approach can be better appreciated by looking at the R-D approximations that are got on real world data. Figure 4.1 illustrates this.

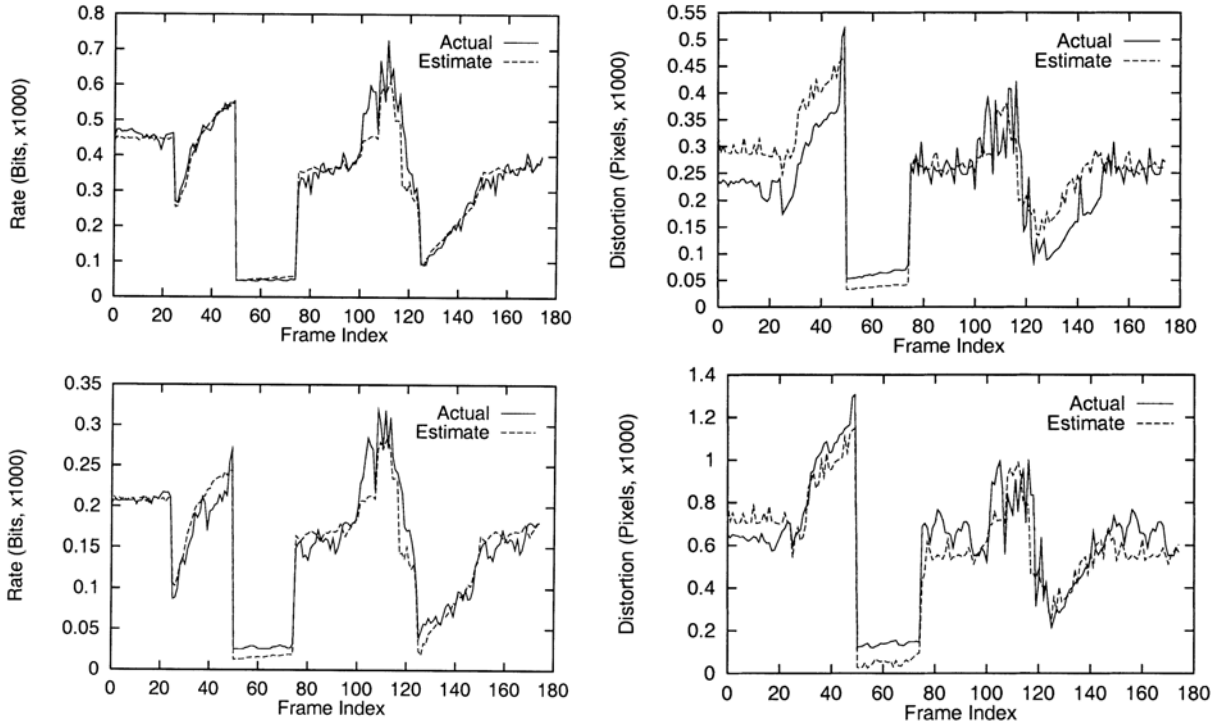

Figure 4.1: The top-left and the bottom-left plots compare the actual rate and the rate estimated by the neural network for half-scale and quarter-scale. The top-right and the bottom-right plots compare the actual and the distortion estimated by the neural network for half-scale and quarter-scale. The first 100 frames were from the training set. Source [9]

The complexity of the above approach makes it suitable only for intra-coded shapes. A simpler linear model [10] has been now proposed for modeling the R-D characteristics. This is based on parameters derived from the border block and block-based shape complexity for the video object. The model being simpler than the neural network based approach can be used in inter and intra modes. This model argues that the bit-rate and the non-normalized distortion increase linearly with the number of border blocks. The rate model is described by

$$\tilde{R}_i = a_i n + b_i$$

where n is the number of the border blocks, and a_i and b_i are model parameters, i denotes the resolution scale, i.e., 0 is full resolution, 1 is CR = 1/2, and 2 is CR = 1/4. Considering the similarity between successive VOPs, linear regression is used to estimate the model parameters as follows:

$$a_i = \frac{\sum_{j=1}^{m} n_j R_{i,j} - \frac{1}{m}(\sum_{j=1}^{m} n_j)(\sum_{j=1}^{m} R_{i,j})}{\sum_{j=1}^{m} n_j^2 - \frac{1}{m}(\sum_{j=1}^{m} n_j)(\sum_{j=1}^{m} n_j)}$$

$$b_i = \frac{1}{m}\sum_{j=1}^{m} R_{i,j} - a_i \frac{1}{m}\sum_{j=1}^{m} n_j$$

where $R_{i,j}$ and n_j are the bit rate and number of the border blocks of m past data points, respectively. On similar lines the distortion model is described by

$$\tilde{D}_i = c_i n + d_i$$

where c_i and d_i are the model parameters which are again found using linear regression on past data points. This is updated in a multiplicative way using a shape complexity measure, as complex shapes tend to produce more distortion than simple ones.
κ defines the shape complexity and is given by

$$\kappa = \frac{\sum_{l=1}^{n} p_l}{S}$$

where p_l denotes the perimeter of each boundary block, n the number of border blocks and S the number pixels in the VOP. The distortion is now defined as

$$\tilde{D}_i = c_i \kappa n + d_i$$

The accuracy of this approach to R-D modeling is confirmed by the simulation results shown in Figure 4.2.

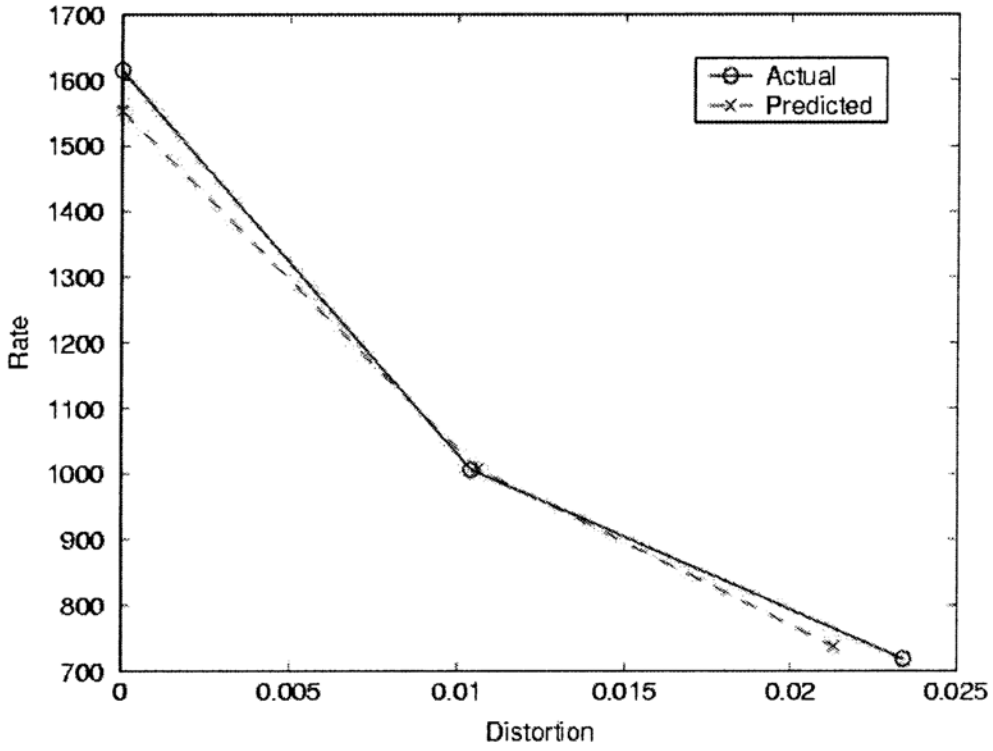

Figure 4.2: Comparison of actual and predicted R-D characteristics for binary shape information got from the linear model. Source [10]

4.2 Rate Control

In shape based representation of video, shape needs to be coded in addition to the texture and allocation of bits for shape coding becomes critical for low bit-rate applications. Shape coding occupies only a small portion of the total bits in high bit-rate applications. This section discusses issues related to rate control for shape coding.

4.2.1 Buffering Policy

Bit-rate is maintained by making sure that on an average the number of bits being encoded is less than a certain buffer threshold. After encoding a frame, the total number of bits used T_c is added to the current buffer level B_c, and decreased from the remaining bits. To ensure that the updated buffer level is not too high, the frame-skip parameter N is set to zero and incremented until the following buffer condition is satisfied

$$B_c < \gamma B_s$$

where B_c is given by

$$B_c = B_p + T_c - T_d(N+1)$$

B_p is the previous buffer level and T_d denotes the number of bits drained at each coding time instant. γ is the skip margin, typically having a value of 0.8. To accommodate the bits required to code shape, the buffer condition is adaptively lowered to

$$B_c + T_{shape} < \gamma B_s$$

where T_{shape} denotes the estimated number of bits required to code the shapes in the next frame. Source [11].

4.2.2 Bit Allocation

The goal of bit allocation is to minimize distortion subject to rate and buffer constraints. The distortion metrics used to quantify distortion in shape and texture coding are different and therefore bit allocation cannot be done jointly. Texture distortion is based on the mean-squared error (MSE) between the original and reconstructed pels, while shape distortion is based on the ratio between the error pels to the total number of nontransparent pels. The problem of bit allocation can be formalized as follows. Given a total rate budget, $R_{total} = R_{texture} + R_{shape}$, how to allocate bits between texture coding and shape coding (note that $R_{texture}$ includes the bits needed for MVs). Since the eyes are more sensitive to distortions in shape than in texture, the problem is tackled by dynamically changing the MPEG-4 parameter AlphaTH (shape rate-control parameter) to meet the maximum shape distortion constraints. Once these constraints are met, the remaining bits are allocated for texture coding. Source [11]

5. Error Control

In general error control in video is done by intra-refresh. The basic idea behind this is to minimize the error propagation by reducing the temporal relation between the frames. This improved error resilience comes at the cost of decreasing coding efficiency, but is still worth doing for applications that demand minimal subjective impact in the presence of transmission errors. For video representation based on objects, both shape and texture data need to be 'refreshed'. Two metrics have been proposed to evaluate the refresh-rate. The first metric called the shape refreshment need (SRN) is defined as the product of the shape error vulnerability and the shape concealment difficulty. The second metric called the texture refreshment need (TRN) is defined as the product of the texture error vulnerability and the texture concealment difficulty. The shape/texture error vulnerability is measured by the fraction of the shape/texture bits that will have to be discarded due to errors in the VOP, while the shape/texture concealment difficulty is measured by the difficulty involved in recovering of shape/texture data from erroneous streams.

A good intra-refresh scheme based on these metrics has been proposed in [12]. In this scheme the contour of the shape is extracted from the binary alpha plane. Missing portions of the contour are then reconstructed by interpolation of available surrounding contours using Bézier curves. The alpha plane is then reconstructed from the fully recovered contour. The data obtained during this process allows the encoder to adaptively determine the rate at which the shape and texture data should be refreshed.

6. Post-Processing

In shape coding different objects are defined for a scene and so composition and alpha blending techniques are required to recreate a scene with the objects. Also errors during transmission of data warrant the need for error concealment. These techniques are discussed in the following sections.

6.1 Composition and Alpha Blending

Lossy coding of shape data can create issues during composition. Undefined pixels or 'holes' may be present in the reconstructed frames at the decoder. One way to fix this issue is to assign some fixed gray value to all the missing pixels. But this does not work well for high levels of distortion. For such cases the undefined pixels are assigned pels from the reconstructed object with minimum distance to the undefined pixel coordinate [2].

6.2 Error concealment

These techniques differ based on whether they use data from previous frames for concealment (temporal) or whether they use information in the current frame only (spatial).

In temporal error concealment, the corrupt data is concealed by copying the co-located block of shape data in the previous frame or by copying the data from the motion compensated block of shape data from the previous frame. These techniques can be further refined assuming that the shape of the object does not change significantly over time. One method could be that the encoder sends the global motion data to the decoder along with the bit stream and this data is then used to fill the missing shape information as before. An improvement over this technique is shown in Figure 6.1.

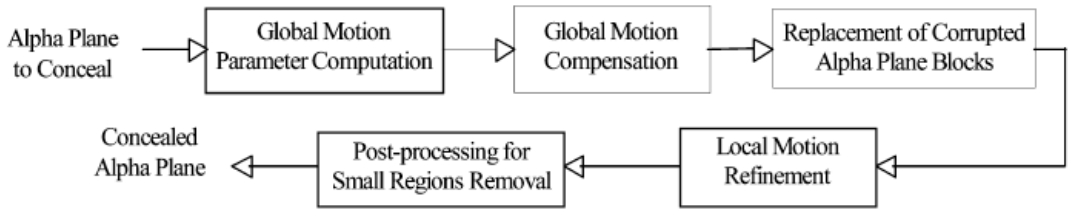

Figure 6.1: Temporal error concealment through computation of global motion parameters. Source [13]

This method eliminates the need to send the global motion data by estimating them at the decoder using the available data. The motion refinement step further improves the performance. The results of this technique are shown in Figure 6.2.

These techniques assume that shape does not change significantly over time. But if it does or if concealment is done for intra-coded shapes then spatial concealment techniques are applied.

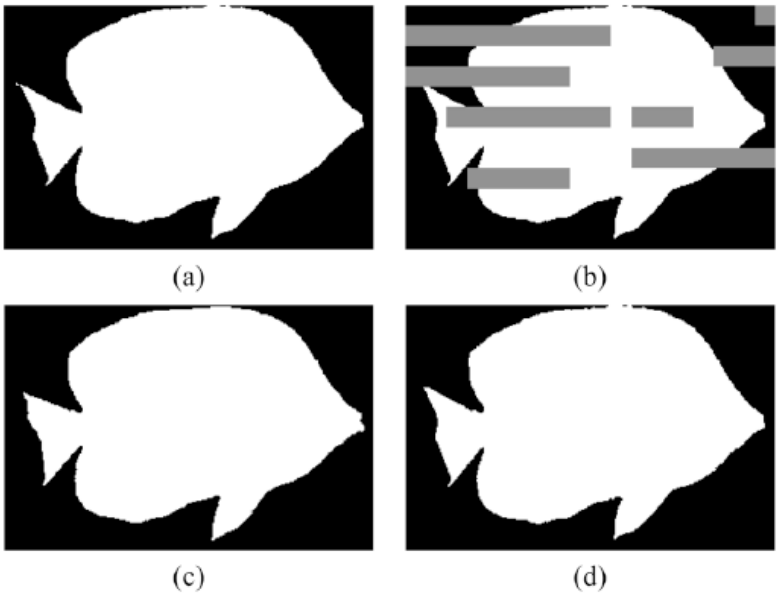

Figure 6.2:.(a) Original uncorrupted alpha plane; (b) corrupted alpha plane; (c) motion compensated previous alpha plane; (d) concealed alpha plane without local motion refinement. Source [13]

One algorithm for spatial concealment [14] computes the missing shape elements as a weighted median of the neighboring shape elements that have been correctly decoded or concealed. The weights are assigned based on the likelihood of an edge in that direction. The algorithm is recursively applied to all the missing blocks. Besides being computationally intensive, the algorithm could also result in concealing of isolated blocks as only local statistics are considered. To overcome these drawbacks an improved algorithm has been proposed [15]. In this scheme the missing contours are reconstructed from available surrounding contours using Bézier curves. Figure 6.3 compares the concealed frames on these two techniques. It is easy to see that the latter method provides fewer artifacts and more accurate shape reconstruction.

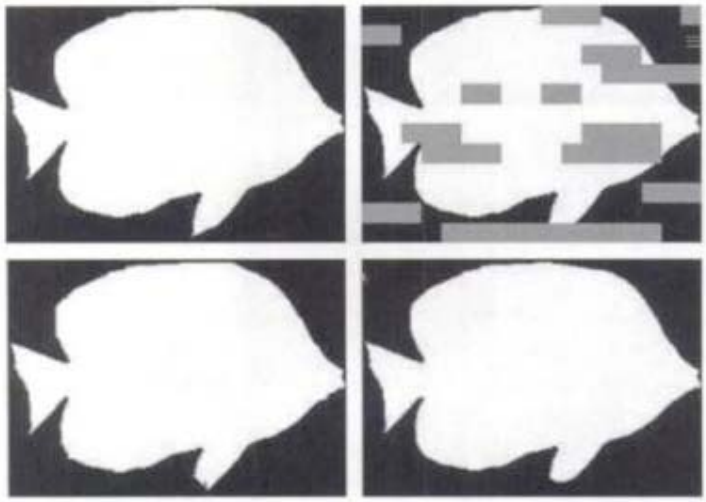

Figure 6.3: Performance comparison of error concealment techniques. Bottom-right is the result of the method described in [15] and bottom-left is from the method described in [14]. Source [2]

7. Applications

7.1 Surveillance

Surveillance applications typically generate huge amounts of video data, all of which need to stored and archived for future access. These systems can tolerate certain inaccuracies in the scene as long as the overall semantics of the scene and the subjective quality of the video are maintained. Object based coding techniques can be used in such cases to achieve efficient storage of video data [16].

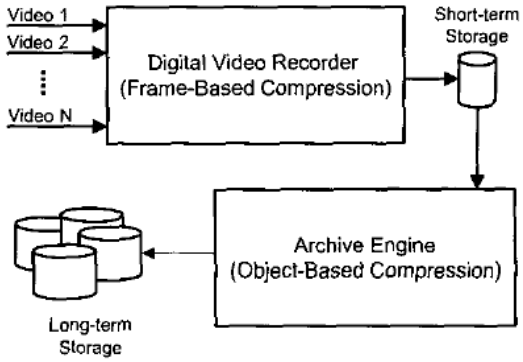

Figure 7.1: Long-term archiving of video surveillance data. Source [16]

Figure 7.1 shows a system that is capable of archiving several months of video content got from several cameras. In this system a single background image is compressed using frame-based coding tools available in MPEG-4. The segmented foreground objects are compressed using object-based coding tools of MPEG-4. As shown in figure 7.2, such a scheme gives rise to differences in background pixels (swaying trees and objects moving due to wind conditions). However the scenes shown in the figure are semantically equivalent and hence good enough for surveillance archiving.

Figure 7.2: Sample reconstructed frames. Left: frame-based reconstruction; Right: object-based reconstruction. Source [2]

Coding of surveillance video using object-based technique results in bit savings between 60 and 90% [16], a significant gain when the amount of data to be stored is considered.

7.2 Interactive TV

Overlaying of a person hand-signing the spoken words over the conventional video content would enable the hearing-impaired to access the information. On similar lines overlaying of news reporters or commentators over live footage at the receiver side (instead of broadcasting) may be needed in applications like internet TV. In both cases, the information to be overlaid can be coded using object-based coding techniques thereby resulting in huge bit savings.

In MPEG-4, shapes can be transmitted independent of the video signals thereby allowing transmission of map of labels. The coordinate of each label can be used to translate a specific request from the user like icons on the PC thereby making TV interactive.

In many multi-view point video conferencing applications, different participants in remote locations need to be presented in a homogenous environment to local participants. The participants can be represented as arbitrarily shaped objects and then composed together over a common background. Since video conferencing systems are allocated lower bandwidths than broadcast video, the bit-savings achieved are hugely beneficial.

Acknowledgement

My main motivation in choosing this topic was to get a deeper understanding of how functionalities in addition to compression are provided in standards to support specialized applications. In this regards I delved more into the realm of shape coding in MPEG-4. I wanted to develop a deep enough understanding of the subject to be able to present this overview. I would like to thank Prof. Jerry Gibson for giving me this opportunity to learn beyond the classroom.

References

[1] Lele Zhou; Zahir, S.; A Novel Shape Coding Scheme For MPEG-4 Visual Standard; ICICIC '06, Vol. 3, 30-01 Aug. 2006, 585 - 588

[2] Mauro Barni; "Document and Image Compression", CRC Press Taylor & Francis Group; 2006 , 299-322

[3] Katsaggelos, A.K.; Kondi, L.P.; Meier, F.W.; Ostermann, J.; Schuster, G.M.; MPEG-4 and rate-distortion-based shape-coding techniques, IEEE Proc. Vol. 86, Issue 6, June 1998, 1126 – 1154

[4] Brady, N.; MPEG-4 standardized methods for the compression of arbitrarily shaped video objects, Circuits and Sys. for Video Tech., IEEE Trans. Vol. 9, Issue 8, 1999, 1170 – 1189

[5] Table B-32 in Annex B of MPEG-4 Visual Specification. ISO/IEC 14496-2:2001. Coding of Audio-Visual Objects-Part 2:Visual, 2d Edition, 2001

[6] I.E.G.Richardson, H.264 and MPEG-4 Video Compression, John Wiley & Sons, 2003, 125-136

[7] F.Pereira, T.Ebrahimi, The MPEG-4 Book, IMSC Press, Prentice Hall, 2002, 318-329

[8] Vetro, A.; Sun, H.; Yao Wang; Guleryuz, O.; Rate-distortion modeling of binary shape using state partitioning, ICIP 99, Vol. 2, 24-28, 1999, 802 - 805

[9] Vetro, A.; Yao Wang; Huifang Sun; Rate-distortion modeling for multiscale binary shape coding based on Markov random fields, Image Proc., IEEE Trans. Vol. 12, Issue 3, 2003, 356 - 364

[10] Zhenzhong Chen; King Ngi Ngan; Linear rate-distortion models for MPEG-4 shape coding, Circuits and Sys. for Video Tech., IEEE. Vol. 14, Issue 6, 2004, 869 – 873

[11] Vetro, A.; Huifang Sun; Yao Wang; MPEG-4 rate control for multiple video objects, Circuits and Sys. for Video Tech., IEEE Trans. Vol. 9, Issue 1, 1999, 186 – 199

[12] Soares, L.D.; Pereira, F.; Spatial shape error concealment for object-based image and video coding, Image Proc., IEEE Trans. Vol. 13, Issue 4, 2004, 586 - 599

[13] Soares, L.D.; Pereira, F.; Temporal shape error concealment by global motion compensation with local refinement, Image Proc., IEEE Trans. Vol. 15, Issue 6, 2006, 1331 – 1348

[14] Shirani, S.; Erol, B.; Kossentini, F.; A concealment method for shape information in MPEG-4 coded video sequences, Multimedia, IEEE Trans. Vol. 2, Issue 3, 2000, 185 – 190

[15] Soares, L.D.; Pereira, F.; Adaptive shape and texture intra refreshment schemes for improved error resilience in object-based video coding, Image Proc., IEEE Trans. Vol. 13, Issue 5, 2004, 662 – 676

[16] Vetro, A.; Haga, T.; Sumi, K.; Sun, H.; Object-based coding for long-term archive of surveillance video. ICME '03, Proc. Vol. 2, 2003, 417-20