12 Two-Dimensional Shape Coding

Joern Ostermann and Anthony Vetro

CONTENTS

12.1 INTRODUCTION

With video being a ubiquitous part of modern multimedia communications, new functionalities in addition to the compression as provided by conventional video coding standards like H.261, MPEG-1, H.262, MPEG-2, H.263, and H.264 are required for new applications. Applications like

content-based storage and retrieval have to allow access to video data based on object descriptions, where objects are described by texture, shape, and motion. Studio and television postproduction applications require editing of video content with objects represented by texture and shape. For collaborative scene visualization like augmented reality, we need to place video objects into the scene. Mobile multimedia applications require content-based interactivity and content-based scalability in order to allocate limited bit rate or limited terminal resources to fit the individual needs. Security applications benefit from content-based scalability as well. All these applications share one common requirement: video content has to be easily accessible on an object basis. MPEG-4 Visual enables this functionality. The main part of this chapter describes MPEG-4 shape coding, the content-based interactivity enabling tool.

Given the application requirements, video objects have to be described not only by texture, but also by shape. The importance of shape for video objects has been realized early on by the broadcasting and movie industries employing the so-called chroma-keying technique, which uses a predefined color in the video signal to define the background. Coding algorithms like object-based analysis-synthesis coding (OBASC) [30] use shape as a parameter in addition to texture and motion for describing moving video objects. Second-generation image coding segments an image into regions and describes each region by texture and shape [28]. The purpose of using shape was to achieve better subjective picture quality, increased coding efficiency as well as an object-based video representation.

MPEG-4 Visual is the first international standard allowing the transmission of arbitrarily shaped video objects (VO) [21]. Each frame of a VO is called video object plane (VOP) consisting of shape and texture information as well as optional motion information. Following an object-based approach, MPEG-4 Visual transmits texture, motion, and shape information of one VO within one bitstream. The bitstreams of several VOs and accompanying composition information can be multiplexed such that the decoder receives all the information to decode the VOs and arrange them into a video scene; the composition of multiple video objects is illustrated in Figure 12.1. Alternatively, objects may be transmitted in different streams according to a scene description [11,44]. This results in a new dimension of interactivity and flexibility for standardized video and multimedia applications.

Two types of VOs are distinguished. For opaque objects, binary shape information is transmitted. Transparent objects are described by gray-scale α-maps defining the outline as well as the transparency variation of an object.

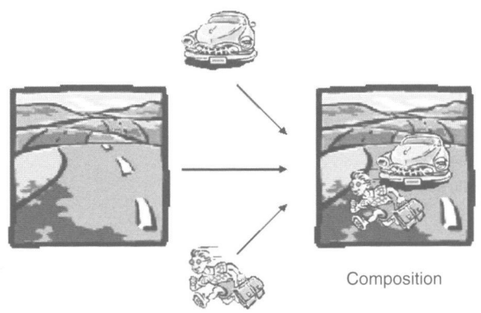

FIGURE 12.1 Content-based scalability requires individual objects to be transmitted and composited at the decoder. Depending on resources, only some of the objects might be composited to the scene and presented at the terminal.

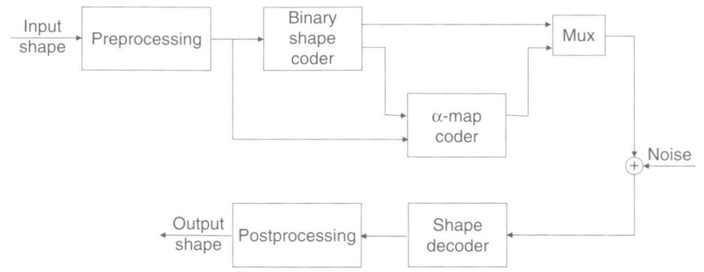

FIGURE 12.2 Processing steps for shape coding considering binary and gray-scale α-maps.

12.1.1 Shape Coding Overview

Figure 12.2 shows the processing steps related to shape coding. They apply to object-based coding systems that transmit shape information only, as well as to systems that transmit texture for the objects. The optional shape preprocessing may remove noise from the shape signal and simplify the shapes such that it can be coded more efficiently. Preprocessing usually depends on the shape coding algorithm employed.

For transparent objects, the preprocessed shape information is separated into a binary shape defining the pels belonging to the object and a gray-scale information defining the transparency of each pel of the object. For binary shapes and gray-scale shape information the binary shape coder codes the shape using lossless or lossy shape coding algorithms. In the case of transparent objects, an α-map coder codes the transparency information for the coded binary shape. The bitstreams get multiplexed, transmitted, and decoded at the decoder. The optional postprocessing algorithm provides error concealment and boundary smoothing.

The receiver decodes the VOs and composes them into a scene as defined by the composition information [11,44]. Typically, several VOs are overlayed on a background. For some applications, a complete background image does not exist. Foreground VOs are used to cover these holes. Often they exactly fit into holes of the background. In case of lossily coded shape, a pixel originally defined as opaque may be changed to transparent, thereby resulting in undefined pixels in the scene. Therefore, lossy shape coding of the background needs to be coordinated with lossy shape coding of the foreground VOs. If objects in a scene are not coded at the same temporal rate and a full rectangular background does not exist, then it is very likely that undefined pixels in the scene will occur. Postprocessing in the decoder may extend objects to avoid these holes.

12.1.2 Related Work

There are three classes of binary shape coders. A *bitmap*-based coder encodes for each pel whether it belongs to the object or not. A *contour*-based coder encodes the outline of the object. In order to retrieve the bitmap of the object shape, the contour is filled with the object label. In the case where there is also texture transmitted with the shape information, an *implicit* shape coder, often referred to as chroma keying [7], can be used, where the shape information is derived from the texture using a predefined color for defining the outside of an object. Similar to texture coding, binary shapes can be coded in a lossless or lossy fashion.

12.1.2.1 Implicit Shape Coding

This class of coding defines a specific pixel value or a range of pixel values as the background of the image. The remaining pels are part of the object. An implicit shape coder is also specified in

GIF89a [9]. For each image, one number can be used to define the value of the transparent pels. All pels of this value are not displayed. Today, GIF89a is used in web applications to describe arbitrarily shaped image and video objects.

12.1.2.2 Bitmap-Based Shape Coding

Such coding schemes are used in the fax standards G4 [4] and JBIG [19]. The modified read (MR) code used in the fax G4 standard scans each line of the document and encodes the location of *changing pels* where the scanline changes its color. In this line-by-line scheme, the position of each changing pel on the current line is coded with respect to either the position of a corresponding changing pel in the reference line, which lies immediately above the present line, or with respect to the preceding changing pel in the current line [31].

12.1.2.3 Contour-Based Shape Coding

Algorithms for contour-based coding and the related contour representations have been published extensively. Different applications nurtured this research: for lossless and lossy encoding of object boundaries, chain coders [12,14] and polygon approximations [15,18,34,42] were developed. For recognition purposes, shape representations like Fourier descriptors were developed to allow translation, rotation, and scale-invariant shape representations [55].

A chain code follows the contour of an object and encodes the direction in which the next boundary pel is located. Algorithms differ by whether they consider a pel having four or eight neighbors for rectangular grids or six neighbors for hexagonal grids. Some algorithms define the object boundary between pels [41]. Freeman [14] originally proposed the use of chain coding for boundary quantization and encoding, which has attracted considerable attention over the last 30 years [23,27,32,38,39]. The curve is quantized using the grid intersection scheme [14] and the quantized curve is represented using a string of increments. Since the planar curve is assumed to be continuous, the increments between grid points are limited to the eight grid neighbors, and hence an increment can be represented by 3 bits. For lossless encoding of boundary shapes, an average 1.2 to 1.4 bits/boundary pel are required [12]. There have been many extensions to this basic scheme such as the generalized chain codes [39], where the coding efficiency has been improved by using links of different length and different angular resolution. In [23], a scheme is presented which utilizes patterns in a chain code string to increase the coding efficiency and in [38], differential chain codes are presented, which employ the statistical dependency between successive links. There has also been interest in the theoretical performance of chain codes. In [27], the performance of different quantization schemes is compared, whereas in [32], the rate–distortion characteristics of certain chain codes are studied. Some chain codes also include simplifications of the contour in order to increase coding efficiency [33,37]. This is similar to filtering the object shape with morphological filters and then coding with a chain code [35]. The entropy coder may code a combination of several directions with just one code word.

A polygon-based shape representation was developed for OBASC [17,18]. As a quality measure, the maximum Euclidean distance $d_{\max}$ between the original and the approximated contour is used. During subjective evaluations of CIF (352×288 pels) video sequences, it was found that allowing a peak distance of $d^*_{\max} = 1.4$ pel is sufficient to allow proper representations of objects in low bit-rate applications. Hence the lossy polygon approximation was developed. Vertices of the spline approximation do not need to be located on the object boundary [24,26].

This polygon representation can be also used for coding shapes in inter mode. 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. It is refined as described above such that the entire polygon is within the allowable error $d^*_{\max}$.

In [22], B-spline curves are used to approximate a boundary. An optimization procedure is formulated for finding the optimal locations of the control points by minimizing the mean-squared error between the boundary and the approximation. In [24], a polygon/spline representation is described which provides optimality in the operational rate distortion sense.

Fourier descriptors describing the contour of an object were developed for applications in recognition, where shape is an important key. Fourier descriptors allow translation, rotation, and scale-invariant representation [36]. In the first step, the coordinates of the contour are sampled clockwise in the xy-plane. This list of 2D coordinates (x_i, y_i) is then transformed into an ordered list $(i, (y_{i+1} - y_i / x_{i+1} - x_i))$ with $0 \leq i \leq i + 1$ being the contour point number and $(y_{i+1} - y_i / x_{i+1} - x_i)$ being the change in direction of the contour. Since the samples are periodic over the object boundary perimeter, they can be expanded into a Fourier series. In order to preserve the main characteristics of a shape, only the large Fourier coefficients have to be maintained. Fourier descriptors are not very efficient in reconstructing polygon-like shapes with only a few coefficients. This is one of the reasons, why they never became very competitive in coding efficiency.

12.1.2.4 MPEG-4 Shape Coding

The MPEG committee investigated implicit, bitmap-based, and contour-based shape coding techniques. Implicit shape coding requires high-quality texture coding in order to derive a high-quality shape. Since humans tend to be more sensitive to shape distortions than to texture distortions, this coupling of shape and texture distortion is not acceptable for many applications. Therefore, while many different approaches for shape coding were evaluated during the development of MPEG-4, proposals for polygon-based contour coding and binary context-adaptive arithmetic coding were the lead contenders. Ultimately, MPEG-4 adopted the binary context-adaptive arithmetic coding, which is elaborated further in the next section.

The publication of the MPEG-4 standard and its work on shape coding [3,26,33] inspired the invention of many shape coding algorithms that are able to outperform MPEG-4 shape coding in terms of coding efficiency while being more computationally demanding. Mainly due to rate distortion optimization, the vertex-based method described in [24] provides an average rate reduction of 7.8% with respect to the content-based arithmetic coder in MPEG-4. The skeleton-based method proposed in [54] gives bit rates 8 to 18% smaller than the MPEG-4 shape coder. In [1], digital straight lines are identified on the contour. These lines are then used to align a template for defining the context of an adaptive arithmetic encoder with the line. Although this algorithm only codes shapes in intra-mode, it requires 33% less bits than the MPEG-4 shape coder in inter-mode as described in Section 12.2.

12.1.3 Chapter Organization

This chapter provides an overview of the algorithms for shape coding in MPEG-4. In Section 12.2, binary and gray-scale shape coding techniques are first described. Then, in Section 12.3, a variety of issues related to the encoding, modeling, and postprocessing of shape are discussed. Section 12.4 presents a few promising application that rely on shape coding, and finally, Section 12.5 presents concluding remarks.

12.2 MPEG-4 SHAPE CODING TOOLS

This section will discuss the coding techniques for 2D shapes using MPEG-4 tools. First, we briefly introduce the general representation for the 2D shape of an object, including binary and gray-scale shape. Then, we describe techniques for coding each of these representations.

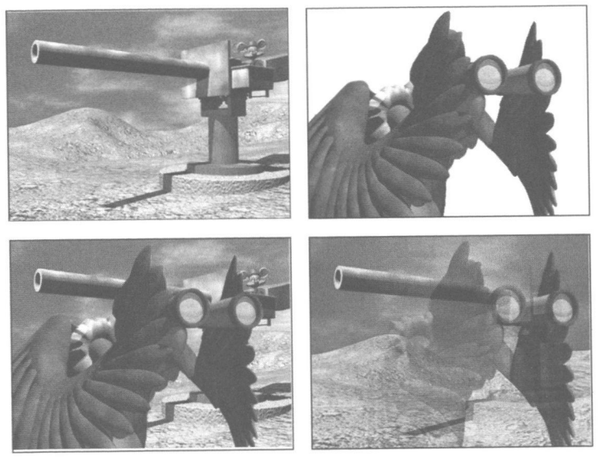

FIGURE 12.3 Comparison of scene composed with background image and foreground object with constant and gray-scale transparency. Top-left: background image; top-right: foreground object; bottom-left: scene composed with constant transparency; bottom-right: scene composed with gray-scale transparency, where the α-map of the foreground object is a horizontal ramp starting with $m_k(0, y) = 0$ on the left side and ending at $m_k(320, y) = 255$ on the right side.

12.2.1 Shape Representation

The 2D shape of an object is defined by means of an α-map M_k of size XY pels:

$$M_k = \{m_k(x, y) \,|0 \le x \le X, 0 \le y \le Y\}, \quad 0 \le m_k \le 255 \tag{12.1}$$

The shape M_k defines for each pel $\mathbf{x} = (x, y)^T$ whether it belongs to the object ($m_k(\mathbf{x}) > 0$) or not ($m_k(\mathbf{x}) = 0$). For an opaque object, the corresponding α-values are 255, and for transparent objects they range from 1 to 255. Usually, the α-map has the same spatial and temporal resolution as the luminance signal of the video sequence. In video-editing applications, the α-map is used to describe object shape and object transparency. Let us assume that we have a background image $s_b(\mathbf{x})$, the object represented by image $s_o(\mathbf{x})$, and the α-map $M_o(\mathbf{x})$. Overlaying the object on the background is done according to

$$s(\mathbf{x}) = \left(1 - \frac{M_o(\mathbf{x})}{255}\right) s_b(\mathbf{x}) + \frac{M_o(\mathbf{x})}{255} s_o(\mathbf{x}) \tag{12.2}$$

As shown in Figure 12.3, the amplitude of the α-map determines how visible the object becomes. We will describe the coding of binary object shapes, i.e., $m_k(\mathbf{x}) \in \{0, 255\}$ in the next subsection, followed by description of the coding of gray-scale shape.

12.2.2 Binary Shape Coding

The MPEG-4 shape coder is known as the context-based arithmetic encoder (CAE) [2,3,24]. It works on macroblocks of size 16×16 pels that the MPEG-4 video encoder defines for a video object. In the following paragraphs, shape encoding in intra-mode is described. Then, this technique is extended to include an inter-mode. The evaluation criteria for lossy binary shape coding are presented last.

12.2.2.1 Intra-Mode

The CAE codes pelwise information only for boundary blocks exploiting the spatial redundancy of the binary shape information to be coded. Pels are coded in scan-line order and row by row. In intra-mode, three different types of macroblocks are distinguished: transparent and opaque blocks are signaled as macroblock type. The macroblocks on the object boundary containing transparent as well as opaque pels belong to the third type. For these boundary macroblocks, a template of 10 pels is used to define the causal context for predicting the shape value of the current pel as shown in Figure 12.4a. For encoding the state transition, a context-based arithmetic encoder is used. The probability table of the arithmetic encoder for the 1024 contexts was derived from several sequences. With 2 bytes allocated to describe the symbol probability for each context, the table size is 2048 bytes.

The template extends up to 2 pels to the left, to the right, and to the top of the pel to be coded. Hence, for encoding the pels in the two top and left rows of a macroblock, parts of the template are defined by the shape information of the already transmitted macroblocks on the top and on the left side of the current macroblock. For the two rightmost columns, each undefined pel of the context is set to the value of its closest neighbor inside the macroblock.

In order to increase coding efficiency as well as to allow lossy shape coding, a macroblock may be subsampled by a factor of 2 or 4 resulting in a subblock of size 8×8 pels or 4×4 pels, respectively. The subblock is encoded using the encoder as described above. The encoder transmits to the decoder the subsampling factor such that the decoder decodes the shape data and then up-samples the decoded subblock to the original macroblock size. Obviously, encoding the shape using a high subsampling factor is more efficient but the decoded shape after up-sampling may or may not be the same as the original shape. Hence, this subsampling is mostly used for lossy shape coding.

Depending on the up-sampling filter, the decoded shape can look somewhat blocky. During the MPEG evaluation process, two filters have been found to perform very well: a simple pel replication filter combined with a 3×3 median filter and slightly better performing adaptive nonlinear up-sampling filter. MPEG-4 decided to adopt the adaptive filter with a context as shown in Figure 12.5 [21]. The value of an up-sampled pel is determined by thresholding a weighted sum of the pels of its context.

The efficiency of the shape coder differs depending on the orientation of the shape data. Therefore, the encoder can choose to code the block as described above or transpose the macroblock prior to arithmetic coding.

Within the MPEG-4 standardization process, the quality of shape coding was controlled by a threshold, AlphaTH, which defines the maximum number of erroneously coded pels in a macroblock without considering effects like change in connectivity, impact on overall object shape, temporal shape

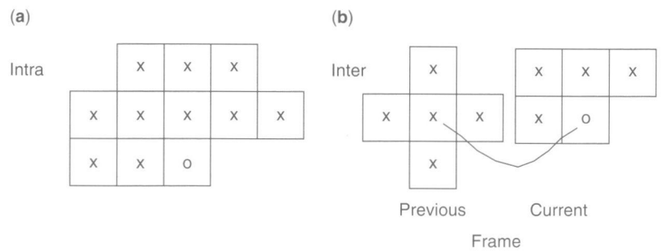

FIGURE 12.4 Templates for defining the context of the pel to be coded (o), where (a) defines the intra-mode context and (b) defines the inter-mode context. The alignment is done after motion compensating the previous VOP.

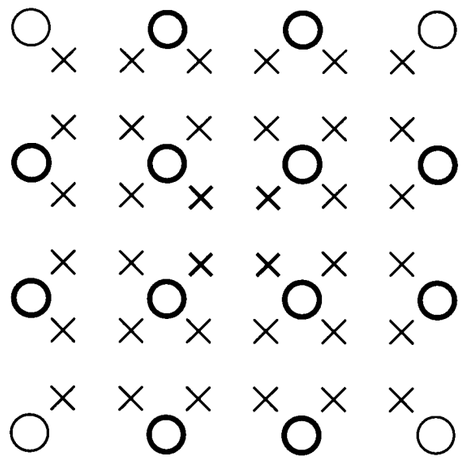

FIGURE 12.5 The up-sampled pels (×) lie between the locations of the transmitted pels (o) from the subsampled macroblock shape information. Neighboring pels (bold o) defining the values of the pels to be up-sampled (bold ×).

variation or the look of the object including its texture. Therefore, lossy shape coding achieves better results if these effects are considered.

12.2.2.2 Inter-Mode

In order to exploit temporal redundancy in the shape information, the coder described above is extended by an inter-mode requiring motion compensation and a different template for defining the context.

For motion compensation, a 2D integer pel motion vector is estimated using full search for each macroblock in order to minimize the prediction error between the previous coded VOP shape M'_{k-1} and the current shape M_k. The shape motion vectors are predictively encoded with respect to the shape motion vectors of neighboring macroblocks. If no shape motion vector is available, texture motion vectors are used as predictors. The shape motion vector of the current block is used to align a new template designed for coding shape in inter-mode as shown in Figure 12.4b.

The template defines a context of 9 pels resulting in 512 contexts. The probability for one symbol is described by 2 bytes giving a probability table size of 1024 bytes. Four pels of the context are neighbors of the pel to be coded, 5 pels are located at the motion-compensated location in the previous VOP. Assuming that the motion vector $(d_x, d_y)^T$ points from the current VOP_k to the previous coded VOP'_{k-1}, the part of the template located in the previously coded shape is centered at $m'_{k-1}(x - d_x, y - d_y)$ with $(x, y)^T$ being the location of the current pel to be coded.

In inter-mode, the same options as in intra-mode are available, such as subsampling and transposing. For lossy shape coding, the encoder may also decide that the shape representation achieved by simply carrying out motion compensation is sufficient thus saving bits by avoiding to code the prediction error. The encoder can select one of the seven modes for the shape information of each macroblock: transparent, opaque, intra, inter with or without shape motion vectors, and inter with or without shape motion vectors and prediction error coding. These different options

with optional subsampling and transposition allow for encoder implementations of different coding efficiency and implementation complexity.

12.2.2.3 Evaluation Criteria for Coding Efficiency

In order to compare the performance of different shape coders, evaluation criteria have to be defined. Within MPEG-4, there are two quality measures for objectively assessing the quality of coded shape parameters. One is the maximum of the minimal Euclidean distance $d^*_{\max}$ (peak deviation) between each coded contour point and the closest contour point on the original contour. This measure allows for an easy interpretation of the shape quality. However, if lossy shape coding results in changing the topology of an object due to opening, closing, or connecting holes, the peak deviation $d^*_{\max}$ is not a useful measure. Therefore, a second measure d_n was used, which is the number of erroneously represented pels of the coded shape divided by the total number of pels belonging to the original shape. Since different objects can have very different ratios of contour pels to interior pels, a given value for d_n only allows to compare with other d_n of different approximations of the same video object. d_n by itself does not provide sufficient information about the shape quality. Hence, some evaluations are done just providing the number of erroneously coded pels.

It was found that the objective measures truthfully reflect subjective quality when comparing different bitmap-based shape coders or when comparing different contour-based shape coders. For lossy shape coding, the bitmap-based shape coders create blocky object shapes whereas contour-based shape coders create an object shape showing polygon edges. Since the two classes of shape coders give different results, a comparison between these two classes has to be done subjectively.

12.2.3 Gray-Scale Shape Coding

Gray-scale α-maps allow 8 bits for each luminance pel to define the transparency of that pel. As shown in Figure 12.3, transparency is an important tool for composing objects into scenes and special effects. Two types of transparencies are distinguished: binary α-maps with objects of constant transparency and arbitrary α-maps for objects with varying transparency.

12.2.3.1 Objects with Constant Transparency

For a transparent object that does not have a varying transparency, the shape is encoded using the binary shape coder and the 8-bit value of the α-map. In order to avoid aliasing, gray-scale α-maps usually have lower transparency values at the boundary. Blending the α-map near the object boundary can be supported by transmitting the coefficients of a 3×3 pel FIR filter that is applied to the α-map within a stripe on the inner object boundary. The stripe can be up to 3 pels wide.

12.2.3.2 Objects with Arbitrary Transparency

For arbitrary α-maps, shape coding is done in two steps [5,7]. In the first step, the outline of the object is encoded as a binary shape. In the second step, the actual α-map is treated like the luminance of an object with binary shape and coded using the MPEG-4 texture coding tools: motion compensation, DCT, and padding. The passing extrapolates the object texture for the background pels of a boundary block.

12.2.4 Texture Coding of Boundary Blocks

For motion-compensated prediction of the texture of the current VOP, the reference VOP is motion-compensated using block motion compensation. In order to guarantee that every pel of the current

VOP has a value to be predicted from, some or all of the boundary and transparent blocks of the reference VOP have to be padded. Boundary blocks are padded using repetitive padding as follows. First, boundary pels are replicated in the horizontal direction, and then they are replicated in the vertical direction. If a value can be assigned to a pel by both padding directions, then an average value is assigned to that pel. Since this repetitive padding puts a significant computational burden on the decoder, a simpler mean padding is used in the second step. Transparent macroblocks bordering boundary blocks are assigned to an average value determined by the pels of its neighboring padded blocks.

In order to encode the texture of a boundary block, MPEG-4 treats the macroblock as a regular macroblock and encodes each block using an 8×8 DCT. The texture is decoded using conventional processing, then discards all information that falls outside of the decoded shape. In order to increase coding efficiency, the encoder can choose the texture of pels outside of the object such that the bit rate is minimized. This non-normative process is also called padding [25]. For intra-mode, a low-pass extrapolation filter was developed, while for inter-mode, setting these pels to 0 was found to perform well in terms of coding efficiency.

12.2.5 Video Coder Architecture

Figure 12.6a shows the block diagram of an object-based video coder [35]. In contrast to the block diagram shown in the MPEG-4 standard, this diagram focuses on the object-based mode in order to allow a better understanding of how shape coding influences the encoder and decoder. Image analysis creates the bounding box for the current VOP s_k and estimates texture and shape motion of the current VOP with respect to the reference VOP s_{k-1}. Shape motion vectors of transparent macroblocks are set to 0. Parameter coding encodes the parameters predictively. The parameters get transmitted and decoded, and the new reference VOP is stored in the VOP memory and also handed to the compositor of the decoder for display.

The increased complexity due to the coding of arbitrarily shaped video objects becomes evident in Figure 12.6b, which shows a detailed view of the parameter coding. The parameter coder first encodes the shape of the boundary blocks using shape and texture motion vectors for prediction. Then, shape motion vectors are coded. The shape motion coder knows which motion vectors to code by analyzing the possibly lossily encoded shape parameters. For texture prediction, the reference VOP is padded as described above. The prediction error is then padded using the original shape parameters to determine the area to be padded. Using the original shape as a reference for padding is again an encoder choice not implemented in the MPEG-4 Reference Software [20] (MoMuSys version). Finally, the texture of each macroblock is encoded using DCT.

12.3 CODEC OPTIMIZATION

The shape coding tools described in the previous subsection provide the basic techniques that are used to efficiently represent shape data. In order to use these tools most effectively, optimizations at various points in the encoder and decoder must be considered. Several key issues are outlined below and discussed further in this section.

Prior to encoding, preprocessing of shape data may be helpful to simplify the information to be coded, while still providing an accurate representation of the object. During the encoding, rate control is needed at the encoder to allocate bits to achieve a maximum quality subject to rate and buffer constraints. Rate–distortion (R–D) models of the shape (and texture) data may be used to alleviate some of the computations involved in making optimal coding decisions. Furthermore, error control may be used to minimize data loss incurred during transmission and error propagation in the reconstructed data.

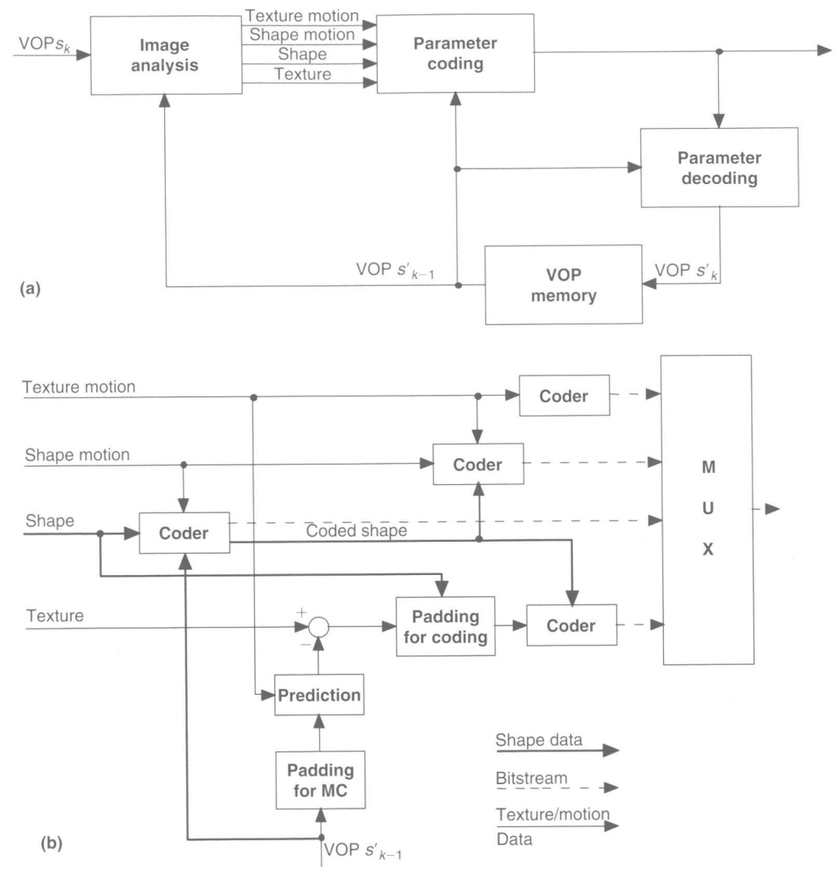

FIGURE 12.6 Block diagram of the video encoder (a) and the parameter coder (b) for coding of arbitrarily shaped video objects.

In addition to the the above encoder issues, postprocessing techniques after decoding also play an important role in reconstructing the object and scene data, and the scene. For one, if errors during transmission have corrupted the reconstructed data, then some form of error concealment should be applied. Also, given multiple objects in a scene, composition of these objects is also required.

12.3.1 Preprocessing

The decoder has no possibility to find out whether the encoder uses lossless or lossy shape coding and what shape coding strategy the encoder uses or padding algorithm for coding is used. In its reference implementation of a video coder, MPEG-4 chose to control lossy shape coding by using an AlphaTH threshold. This threshold defines the maximum number of incorrectly coded pels within a boundary

block and allows the topology of the shape to change. Often, isolated pels at the object boundary are coded as part of the object. Using morphological filters to smooth object boundaries provide a much more predictable quality of a lossily encoded shape [35].

12.3.2 Rate–Distortion Models

For texture coding, a variety of models have been developed that provide a relation between the rate and distortion, e.g., [8,16]. These models are most useful for rate control of texture information. Given some bit budget for an object or frame, one can determine a quantizer value that meets a specified constraint on the rate. Additionally, such models can be used to analyze the source or sources to be encoded in an effort to optimize the coding efficiency in a computationally efficient way.

In the case of shape coding, analogous models have been developed, mainly for use in the analysis stage of an object-based encoder. The primary motivation to develop such models is to avoid performing actual coding operations to obtain the *R–D* characteristics of the binary shape. In the case of MPEG-4 shape coding, this involves down-sampling and arithmetic coding operations to obtain the rate, and up-sampling and difference calculations to compute the distortion at various scales. Figure 12.7 shows sample images for binary shapes at different resolutions and after up-sampling. Accurate estimates of the *R–D* characteristics can play a key role in optimizing the bit

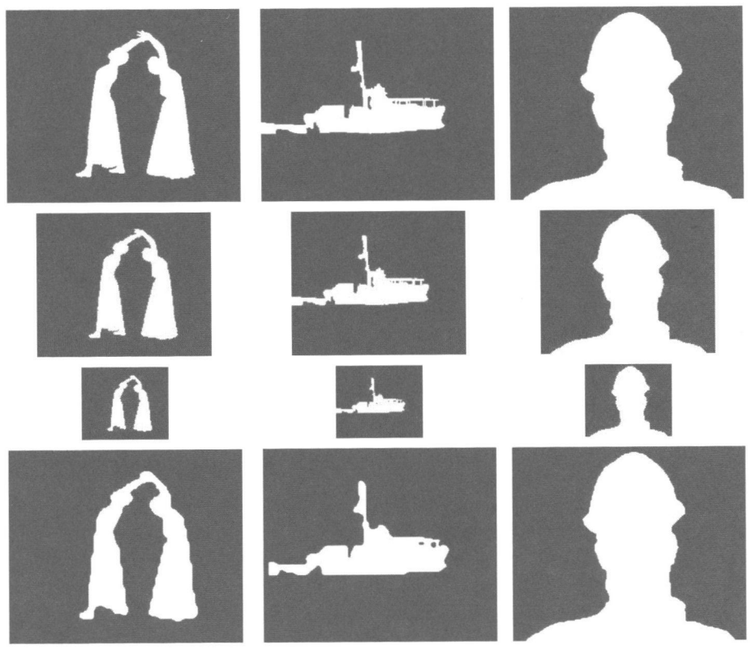

FIGURE 12.7 Sample test shapes from the *Dancer*, *Coastguard*, and *Foreman* sequences. The first row shows the original shapes at full resolution. The second and third rows show the shapes down-sampled by factors 2 and 4 according to the MPEG-4 standard. The bottom row shows the up-sampled reconstructed shapes from quarter-scale images (*Dancer* and *Coastguard*) and half-scale image (*Foreman*) according to the MPEG-4 standard.

allocation for the binary shape among blocks and between shape and texture coding with much lower complexity. Also, knowing the expected rate for shape can stabilize the buffer occupancy of a video coder, especially for low bit-rate video coding [50].

In the following, we review several approaches that have been proposed for modeling the *R–D* characteristics of binary shape and discuss the performance of each. Throughout this section, we aim to model the characteristics of the MPEG-4 CAE-based shape coder described in Section 12.2. Also, we assume shapes are coded in intra-mode only, unless otherwise specified.

For the purpose of this subsection, the modeling problem is formally stated as follows. Let $(R, D)_k$ denote the *R–D* values for a binary shape that is coded at resolution k. Using input parameters, θ_i, which represent features extracted from the shape data, and modeling function, $f(\cdot)$, we consider the approximation,

$$(R, D)_k \approx f(\theta_i) \tag{12.3}$$

The first attempt to solve this problem attempted to categorize all possible binary patterns over a 2×2 neighborhood into N states [52]. This approach was referred to as state partitioning. As expected, the rate is predicted well by this approach, but the distortion suffers from large prediction error. The reason for this is because the rate could be accurately modeled from a fixed 2×2 neighborhood, such that the 10-bit states used by CAE can be correctly collapsed into one of the available states. The distortion, on the other hand, is not modeled so accurately because the actual up-sampling process uses a 12-pel neighborhood and estimating distortion based on the 2×2 pixels is not sufficient.

An improved probabilistic approach for modeling the *R–D* characteristics of binary shape was proposed in [53]. In this work, the shape model is based on the statistics (or moments) that one can extract from the data. With this approach, the aim was to have a distribution whose samples resemble the type of data that we are trying to code. At the same time, this model should be able to make distinctions between different shapes at the same scale, and also between the same shape at different scales.

In [10], a Markov random field (MRF) model that is able to represent the fine structures of an image is presented. The model relies on three parameters: edge, line, and noise. It has been shown in [53] that the moments of this model, which are considered sufficient statistics of the distribution, exhibit favorable properties for modeling as outlined above.

To estimate the rate and distortion at various scales, a simple multilayer feed-forward network has been proposed in [53], where the input to this network are the statistical moments of the MRF model that are calculated from the original shape image, and the output is the estimated rate and distortion of the shape data at different scales.

The plots in Figure 12.8 provide a comparison of the actual rates and output rates of the neural network for each of the three scales. In these plots, the first 100 frames correspond to the training set and the next 75 frames correspond to the testing set. We can see that that the neural network does indeed provide close estimates to the actual rate and follows the trends quite well for both the training and testing data. Similarly, the plots in Figure 12.9 provide a comparison of the actual distortions and output distortions of the neural network for the reconstructed binary maps from half and quarter scales. It should be noted that the absolute errors plotted in Figure 12.9 vary significantly in some cases due to size or complexity of the shape boundary. If these errors were normalized to the object size, the relative errors would be much closer.

While the results shown here demonstrate that MRF parameters can be used to model multiscale rate–distortion characteristics of binary shape, and do so with information provided at the full resolution only, the above method has only been applied to intra-coded shapes. In [6], a linear *R–D* model has been proposed based on parameters derived from the boundary blocks and a block-based shape complexity for the video object. Besides being much simpler than a neural network-based prediction, the linear model is applied to both intra- and inter-coded shape.

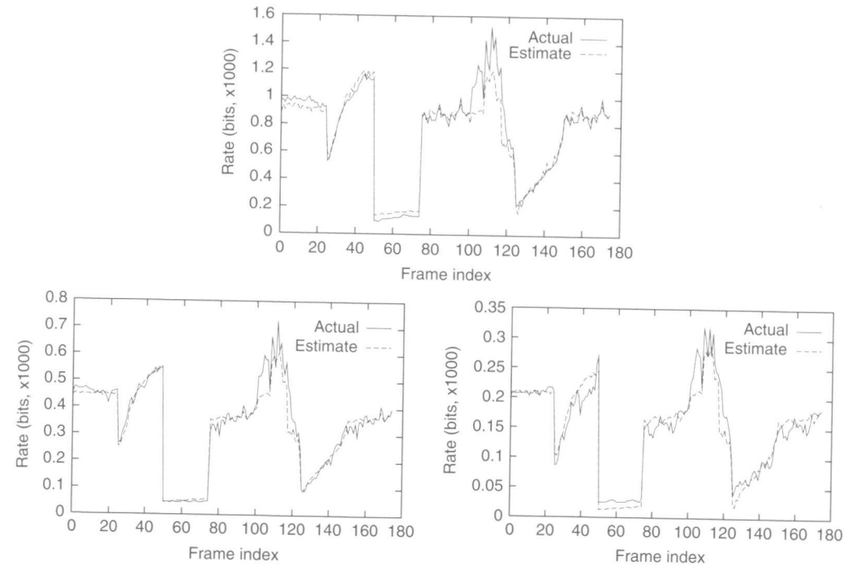

FIGURE 12.8 Comparison of actual rates and output rates of neural network from [53] at full-scale (top), half-scale (bottom-left), and quarter-scale (bottom-right). The first four clips (corresponding to the first 100 frames) are part of the training set, while the following three clips (corresponding to the remaining 75 frames) are part of the test set.

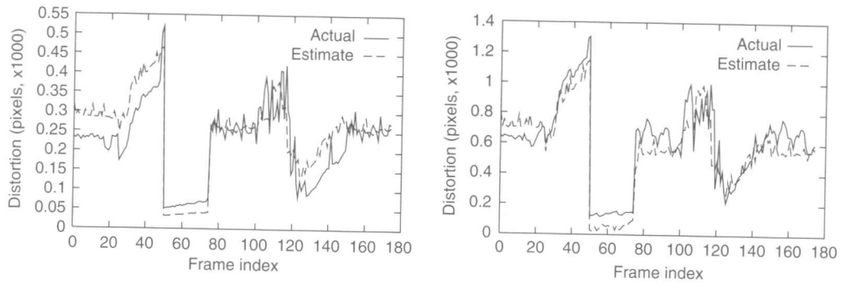

FIGURE 12.9 Comparison of actual distortions and output distortions of neural network from [53]. Distortion is measured between original and reconstructed binary map from half-scale (left) and quarter-scale (right). The first four clips (corresponding to the first 100 frames) are part of the training set, while the following three clips (corresponding to the remaining 75 frames) are part of the test set.

As stated in [6], the rate model is given by

$$\tilde{R}_i = a_i n + b_i \tag{12.4}$$

where n is the number of boundary blocks, a_i and b_i are model parameters, and i denotes the resolution scale. Linear regression is used to estimate the model parameters based on past data points.

Similar to the rate model, the distortion model is given by

$$\tilde{D}_i = c_i n + d_i \tag{12.5}$$

where c_i and d_i are model parameters, which are also calculated using linear regression on past data points. However, it has been observed that more complex boundaries will always produce more distortion, therefore a shape complexity measure based on a normalized perimeter has been introduced:

$$\kappa = \frac{\sum_{l=1}^{n} p_l}{S} \tag{12.6}$$

where p_l denotes the perimeter of each boundary block and S the number of nontransparent pixels in the shape. This complexity measure is factored into the distortion model in a multiplicative way such that the new distortion model becomes

$$\tilde{D}_i = c_i \kappa n + d_i \tag{12.7}$$

Simulation results in [6] confirm the accuracy of the above approach for both intra- and inter-coded shapes at various levels of resolution.

12.3.3 Rate Control

In object-based coding, the problem of coding additional shape information becomes critical at lower bit rates. At higher bit rates, the shape bits occupy a much smaller percentage of the overall rate. In the following, we first describe a buffering policy to account for the additional shape information in object-based coding. With object-based coding, there is also some flexibility in allocating rate among texture and shape information, so we also discuss bit allocation among texture and shape. In both cases, the bits used for shape are estimated from the *R–D* models described in Section 12.3.2. Finally, we discuss how rate control decisions for different objects may impact the overall composition of a scene.

12.3.3.1 Buffering Policy

In frame-based rate control, the bits used to encode a frame, T_c, are added to the buffer. Let the buffer occupancy at time t be denoted by $B(t)$. To help ensure that the buffer will not overflow, a frameskip parameter, N, is set to zero and incremented until the following buffer condition is satisfied:

$$B(t) < \gamma B_s \tag{12.8}$$

where B_s is the size of the buffer, and the value of γ denotes a buffer margin having a typical value of 0.8. In the above, the updated buffer level is given by

$$B(t) = B(t - \tau) + T_c - T_d(N + 1) \tag{12.9}$$

where $B(t - \tau)$ denotes the buffer level of the previously coded frame and T_d is the amount of bits drained from the buffer at each coding time instant.

In object-based coding, the buffering policy must account for the high percentage of shape bits at low bit rates. Let the estimated number of bits to code the shape for all objects in the next frame be denoted by T_{shape}. Then, to ensure that there are a sufficient number of bits to code the texture in the next frame, the buffer condition given by Eq. (12.8) is slightly modified as

$$B(t) + T_{shape} < \gamma B_s \tag{12.10}$$

Another way to view the modified buffer condition is that the buffer margin is adaptively lowered according to the estimated bits for shape. In this way, if a relatively large number of bits are estimated for the shape, the rate control can account for these additional bits by skipping frames and allowing the buffer to drain. This will allow a sufficient number of bits to code all the information in the next frame, without compromising the spatial quality more than necessary.

12.3.3.2 Bit Allocation

Generally speaking, bit allocation aims to minimize distortion subject to rate and buffer constraints. In an object-based coding framework, allocation of bits used for both shape and texture data could be performed in a joint manner. However, since the distortion metrics for shape and texture are different, it is not straightforward to derive a single cost function that accounts for both metrics. It should be noted that binary shape distortion is usually defined as the ratio of error pixels to the total number of nontransparent pixels, while texture distortion is defined according to mean-squared error between original and reconstructed pixel values.

Let the total rate budget for an object be given by $R = R_t + R_s$, where R_t are the bits allocated for texture (including motion) and R_s are the bits allocated for shape. Given this allocation, the optimal texture and shape coding modes for each block in the object could be determined separately using, for example, conventional Lagrangian multiplier techniques. If *R–D* models are used to estimate the operating points for texture and shape at various quantizer values and conversion ratios, respectively, then some of the computational burden could be alleviated. The problem is then reduced to one of allocating the total bits for an object among R_t and R_s. This problem may also be posed at the frame level in which the optimization is conducted for all objects in a scene.

The main difficulty with the above approach is allocating bits between shape and texture. To overcome this problem, a practical approach has been proposed in [50], where rather than allocating a specific number of bits to shape, the maximum distortion for shape is controlled through the AlphaTH threshold instead. In this way, the shape will consume some portion of the total rate budget for a frame subject to a maximum distortion constraint and the remaining bits would be allocated to the texture. In [50], a method to dynamically adapt the AlphaTH threshold according to the buffer fullness and quality of the texture encoding has been proposed.

12.3.4 Error Control

It is largely recognized that intra-refresh can play a major role in improving error resilience in video coding systems. This technique effectively minimizes error propagation by removing the temporal relation between frames. Although the intra-refresh process decreases the coding efficiency, it will significantly improve error resilience at the decoder, which increases the overall subjective impact in the presence of errors in the transmission.

A key aspect to consider in applying intra-refresh schemes is to determine which components of the video to refresh and when. For object-based video, both shape and texture data must be considered. In [45], shape refreshment need (SRN) and texture refreshment need (TRN) metrics have been proposed. The SRN is defined as a product of the shape error vulnerability and the shape concealment difficulty, while the macroblock-based TRN is similarly defined as a product of the texture error vulnerability and the texture concealment difficulty. In both cases, the error vulnerability measures the statistical exposure of the shape/texture data to channel errors and the concealment difficulty expresses how difficult the corrupted shape/texture data is to recover when both spatial and temporal error concealment techniques are considered.

Based on the above refreshment need metrics, novel shape and texture intra-refreshment schemes have been proposed in [46]. These schemes allow an encoder to adaptively determine when the shape and texture of the various video objects in a scene should be refreshed in order to maximize the decoded video quality for a certain total bit rate.

12.3.5 Post-Processing

Two specific needs for postprocessing shape data are considered in this subsection. First, given that the shape data defines objects in a scene, composition and alpha-blending techniques are required to generate a complete scene consisting of multiple objects. Second, considering that errors may occur during transmission, methods for error concealment become necessary. Both of these operations are performed at the receiver and will be discussed further below.

12.3.5.1 Composition and Alpha Blending

As mentioned earlier, composition issues may arise due to lossy coded shape or the coding of multiple objects at different temporal rates. In [50], undefined pixels due to lossy shape coding were simply assigned a gray value. Since maximum distortion in a block was controlled using AlphaTH, which only took values in the range [0, 64], the impact on visual quality was not noticeable. With higher levels of shape distortion, more sophisticated methods would be required to recover the pixel values for the texture. In [29], object coding with different temporal rates was considered. In this work, undefined pixels were assigned values from the reconstructed object with minimum distance to the undefined pixel coordinate.

12.3.5.2 Error Concealment

Concealment techniques could be divided into two categories: temporal and spatial. Temporal concealment techniques rely on shape information from previous time instants to do the concealment, while spatial concealment techniques use information only from the current time instant. Several techniques for spatial and temporal concealment are outlined below.

The earliest known attempt to address this concealment problem for shape was proposed in [13], where the idea of extending conventional motion-compensated concealment techniques for texture to shape was explored. In this way, when a given block of shape data is corrupted, the decoder conceals it by copying a block of shape data from the previous time instant. This could be achieved by simply copying the co-located block of shape data from the previous time instant. Alternatively, a motion vector estimate may be obtained such that the corrupted block in the current time instant is replaced by a displaced shape block from the previous time instant. Temporal concealment was further investigated in [40] considering that the boundary of the shape data of a given video object does not change significantly in time and thus these changes can be described by a global motion model. Based on this assumption, the global motion parameters are estimated at the encoder and sent along with the encoded bitstream to the decoder. With this method, the decoder would apply global motion compensation using parameters derived at the encoder to restore the corrupted contour, then fill in the concealed contour to recover the entire shape. An extension of this approach was proposed in [48], which eliminates the need for an encoder to send additional information by performing the estimation of global motion parameters at the decoder using available shape data. Additionally, this approach improves performance by adding a motion refinement step to better deal with shapes that have some local motion.

The above temporal concealment techniques are advantageous since there is access to past information that can significantly enhance the recovery of shape data, especially in cases where the shape does not change much. However, when the shape changes significantly over time or concealment is applied to still or intra-coded shapes, spatial concealment techniques should be used instead.

Spatial shape concealment was first addressed in [43]. In this work, the shape was modeled with a binary MRF and concealed based on maximum *a posteriori* (MAP) estimation. According to this algorithm, each missing shape element in a missing block is estimated as a weighted median of the neighboring shape elements that have been correctly decoded or concealed, where the weights are assigned adaptively based on the likelihood of an edge in that direction. For each missing shape

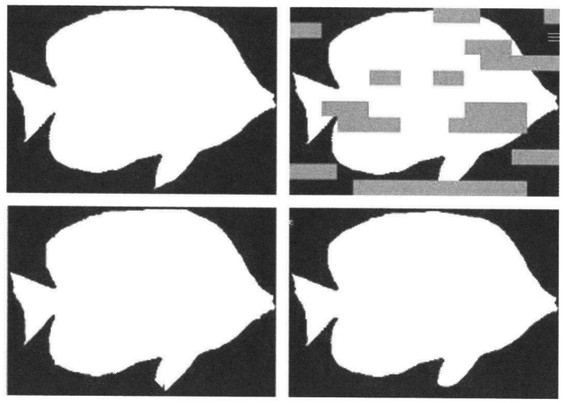

FIGURE 12.10 Performance comparison of error concealment techniques. Top-left: original shape image; top-right: corrupted shape image with 25% of total blocks lost; bottom-left: shape image recovered using method of Shirani et al. [43]; bottom-left: shape image recovered using method of Soares and Pereira [47].

block, this procedure is iteratively repeated until the algorithm converges. Additionally, if several consecutive missing blocks have to be concealed, the procedure is recursively applied to all the missing shape blocks. This method obviously has some computational burdens, but may also suffer in concealing isolated blocks since only local statistics of the data are considered. In an attempt to overcome these drawbacks, a method that interpolates the missing contours based on the available surrounding contours using Bezier curves has been proposed in [47]. These cubic parametric curves are fully determined by four points, which include two end points at the boundaries of a lost block and two additional control points within the region of the lost block. A method for determining the control points that ensure certain smoothness and continuity constraints has been presented. Figure 12.10 compares the performance between the concealment method of Shirani et al. [43] and that of Soares and Pereira [47]. Besides providing a lower complexity solution, the method by Soares and Pereira provides an improved recovery of the shape data that is characterized by a more accurate representation of the original shape and fewer artifacts.

12.4 APPLICATIONS

This section describes how shape coding and object-based video coding, in general, could be applied for surveillance and interactive TV applications.

12.4.1 Surveillance

Here we describe a surveillance application system that utilizes object-based coding techniques to achieve efficient storage of video content [49]. In this system, certain inaccuracies in the reconstructed scene can be tolerated; however, subjective quality and semantics of the scene must be strictly maintained. As shown in Figure 12.11, the target of this system is the long-term archiving of surveillance video, where several months of video content from multiple cameras would need to be stored.

One of the major advantages of object-based coding is that each object can vary in its temporal quality. However, in terms of rate–distortion metrics, only minimal gains in coding efficiency could be achieved [51] since the composition problem typically limits the amount of temporal frame skipping

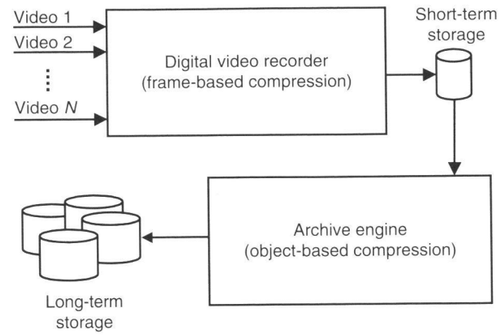

FIGURE 12.11 Application system for surveillance employing object-based coding for long-term archive of video content.

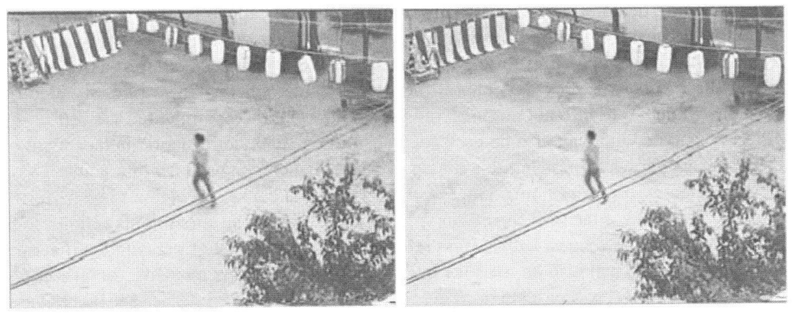

FIGURE 12.12 Sample reconstructed frame of *festA* sequence. Left: frame-based reconstruction; right: object-based reconstruction.

of the background object. Fortunately, in the surveillance system being discussed here, obtaining the full background without any foreground objects is not a problem. Also, subjective video quality is the main consideration.

In this system, a single background image is compressed using frame-based coding, and the sequence of segmented foreground objects are compressed using object-based coding; the background image is simply repeated for each reconstructed frame. Performing the object-based compression in this way gives rise to differences in the background pixels, especially for outdoor scenes that, for example, have swaying trees and objects due to wind conditions.

To demonstrate the effectiveness of this approach, the above object-based coding methodology is applied to several surveillance test sequences. Sample reconstructed frames of one sequence using frame- and object-based coding are shown in Figure 12.12. In this test sequence, wind is blowing the striped curtain, tree branches, and hanging ornaments; all of which are coded and accurately represented by the frame-based coding result. However, for the object-based coding result, these moving background elements are not recorded and only the moving foreground object is coded at each time instant. Semantically, however, these sample frames are equivalent. Table 12.1 summarizes the

TABLE 12.1
Comparison of Storage Requirements (in KB) for Frame-Based and Object-Based Coding

Sequence	festA	festB	festC	festD	rain	tree
Frame-based	178	173	35	116	78	43
Object-based	17	18	13	18	22	10
% Savings	90.5	89.6	62.9	84.5	71.8	76.7

Note: Background image for object-based coding results are included (fest: 9KB; rain: 4KB; tree: 4KB).

storage requirements over a broader range of sequences; it is clear that object-based coding provides favorable savings in the bits required to store the compressed video sequences.

12.4.2 Interactive TV

With the increasing demand for access to information for handicapped people, TV has to become accessible for the hearing impaired. Overlaying a person signing the spoken words over the video will enable this functionality. Transmitting this person using object-based coding and overlaying this object over the regular TV program enables the user to select whether this optional video object should be presented in order to not diminish the presentation for the nonimpaired viewers.

MPEG-4 shape coding can also be used for transmitting shape information independent of the video signal. This allows for transmitting a map of labels that can be used for interactive TV. If the user moves a cursor with the remote control over the video and clicks, the action to be performed is determined by the label of the map corresponding to the position of the cursor. This action can select items for teleshopping or request background information on the object or person identified.

Object-based video can also be used for overlaying news reporters over live footage at the TV receiver instead of at the broadcast studio. This is of special interest for Internet TV, where the same news story might require reporters talking in different languages. In this scenario, the background video is sent as an rectangular-shaped video object on one channel and the news reporter with the desired language represented as an arbitrarily shaped video object is selected on the second channel. The receiver composes these objects into one video.

Multipoint video conference systems would like to present the remote participants in an homogeneous environment to the local participant. This can be easily achieved when each participant is represented as an arbitrarily shaped video object. Scene composition can arrange the video objects on a common background and present the result to the local participant.

12.5 CONCLUDING REMARKS

This chapter has described the shape coding tools that have been adopted in the MPEG-4 coding standard. The context-based arithmetic encoder uses a template to define the context for coding the current pel of a binary shape bitmap. If objects have arbitrary transparency, then these values are coded using MPEG-4 tools like DCT and motion compensation. The integration of a shape coder with texture coding for object-based video coding requires use of texture extrapolation also known as padding. Lossy shape coding may create pels with undefined texture requiring the use of similar extrapolation techniques as well as concealment techniques. For video coding, binary shape coding is enabled in the MPEG-4 core, main and enhanced coding efficiency profiles, while gray-scale shape coding is only enabled in the main and enhanced coding efficiency profiles.

Beyond the fundamental shape coding techniques, we have discussed several issues related to encoder optimization, including *R–D* modeling of binary shape, rate control, and error control. A linear *R–D* model has shown to be effective in predicting rate and distortion of shape at various resolutions. Based on such models, we have reviewed buffering policies for maintaining a stable buffer as well as bit allocation techniques to distribute the rate among shape and texture. Furthermore, error control techniques that attempt to minimize error propagation in the decoder based on refreshment need metrics have been described.

Postprocessing of shape data has also been covered in this chapter. In particular, an overview of various error-concealment techniques for the recovery of lost data during transmission has been described. Temporal concealment techniques are advantageous since there is access to past information to improve recovery results; however, spatial concealment is still needed in cases where the shape changes significantly over time or for concealment of still or intra-coded images.

Finally, two promising applications of shape coding have been presented, including the use of object-based coding for long-term archiving of surveillance video resulting in bit savings between 60 and 90%, and interactive television.

REFERENCES

1. S.M. Aghito and S. Forchhammer, Context based coding of binary shapes by object boundary straightness analysis, *IEEE Data Compression Conference*, UT, USA, March 2004.
2. N. Brady, MPEG-4 standardized methods for the compression of arbitrarily shaped objects, *IEEE Trans. Circuits Syst. Video Technol.*, 9 (1999) 1170–1189.
3. N. Brady and F. Bossen, Shape compression of moving objects using context-based arithmetic coding, *Signal Process. Image Commun.*, 15 (2000) 601–618.
4. CCITT, Facsimile Coding Schemes and Coding Functions for Group 4 Facsimile Apparatus, CCITT Recommendation T.6, 1994.
5. W. Chen and M. Lee, Alpha-channel compression in video coding, *ICIP 97*, Special session on shape coding, Santa Barbara, 1997.
6. Z. Chen and K. Ngan, Linear rate-distortion models for MPEG-4 shape coding, *IEEE Trans. Circuits Syst. Video Technol.*, 14 (2004) 869–873.
7. T. Chen, C.T. Swain, and B.G. Haskell, Coding of subregions for content-based scalable video, *IEEE Trans. Circuits Syst. Video Technol.*, 7 (1997) 256–260.
8. T. Chiang and Y.-Q. Zhang, A new rate control scheme using quadratic rate-distortion modeling, *IEEE Trans. Circuits Syst. Video Technol.*, 7 (1997) 246–250.
9. Compuserve, *Graphics Interchange Format (sm), Version 89a*, CompuServe Incorporated, Columbus, OH, July 1990.
10. X. Descombes, R.D. Morris, J. Zerubia, and M. Berthod, Estimation of Markov random field parameters using Markov chain Monte Carlo maximum likelihood, *IEEE Trans. Image Process.*, 8 (1999) 954–962.
11. J.-C. Duford, BIFS: scene description, in *The MPEG-4 Book*, T. Ebrahimi and F. Pereira, Eds., IMSC Press Multimedia Series, Prentice-Hall, Upper Saddle River, 2002, pp. 103–147.
12. M. Eden and M. Kocher, On the performance of a contour coding algorithm in the context of image coding. Part I: contour segment coding, *Signal Process.*, 8 (1985) 381–386.
13. M.R. Frater, W.S. Lee, M. Pickering, and J.F. Arnold, Error concealment of arbitrarily shaped video objects, *Proceedings of IEEE International Conference on Image Processing*, vol. 3, pp. 507–511, Chicago, USA, Oct. 1998.
14. H. Freeman, On the encoding of arbitrary geometric configurations, *IRE Trans. Electron. Comput.*, EC-10 (1961) 260–268.
15. P. Gerken, Object-based analysis-synthesis coding of image sequences at very low bit rates, *IEEE Trans. Circuits Syst. Video Technol.*, 4 (1994) 228–235.
16. H.M. Hang and J.J Chen, Source model for transform video coder and its application – Part I: fundamental theory, *IEEE Trans. Circuits Syst. Video Technol.*, 7 (1997) 287–298.

17. M. Hoetter, Optimization and efficiency of an object-oriented analysis-synthesis coder, *IEEE Trans. Circuits Syst. Video Technol.*, 4 (1994) 181–194.
18. M. Hötter, Object-oriented analysis-synthesis coding based on two dimensional objects, *Signal Process.: Image Commun.*, 2 (1990) 409–428.
19. ISO/IEC JTC1/SC29/WG1, IS11544 — Coded Representation of Picture and Audio Information Progressive Bi-Level Image Compression, ISO, 1992.
20. ISO/IEC, JTC1/SC29/WG11, IS14496-5 — Information Technology — Coding of Audio-Visual Objects — Part 5: Reference Software, ISO, 1999.
21. ISO/IEC, JTC1/SC29/WG11, IS14496-2 — Information Technology — Coding of Audio-Visual Objects — Part 2: Visual, 3rd ed., ISO, 2004.
22. A.K. Jain, *Fundamentals of Digital Image Processing, Prentice-Hall*, Englewood Cliffs, NJ, 1989.
23. T. Kaneko and M. Okudaira, Encoding of arbitrary curves based on the chain code representation, *IEEE Trans. Commun.*, 33 (1985) 697–707.
24. A. Katsaggelos, L.P. Kondi, F.W. Meier, J. Ostermann, and G.M. Schuster, MPEG-4 and rate-distortion-based shape-coding techniques, *Proc. IEEE*, 86 (1998) 1126–1154.
25. A. Kaup, Object-based texture coding of moving video in MPEG-4, *IEEE Trans. Circuits Syst. Video Technol.*, 9 (1999) 5–15.
26. J.I. Kim, A.C. Bovik, and B.L. Evans, Generalized predictive binary shape coding using polygon approximation, *Signal Process.: Image Commun.*, 15 (2000) 643–663.
27. J. Koplowitz, On the performance of chain codes for quantization of line drawings, *IEEE Trans. Pattern Anal. Machine Intell.*, 3 (1981) 180–185.
28. M. Kunt, A. Ikonomopoulos, and M. Kocher, Second-generation image-coding techniques, *Proc. IEEE*, 73 (1985) 549–574.
29. J.-W. Lee, A. Vetro, Y. Wang, and Y.-S. Ho, Bit allocation for MPEG-4 video coding with spatio-temporal trade-offs, *IEEE Trans. Circuits Syst. Video Technol.*, 13 (2003) 488–502.
30. H.G. Musmann, M. Hötter, and J. Ostermann, Object-oriented analysis-synthesis coding of moving images, *Signal Process.: Image Commun.*, 1 (1989) 117–138.
31. A.N. Netravali and B.G. Haskell, *Digital Pictures — Representation and Compression*, Plenum Press, New York, 1988.
32. D.L. Neuhoff and K.G. Castor, A rate and distortion analysis of chain codes for line drawings, *IEEE Trans. Inf. Theory*, 31 (1985) 53–67.
33. P. Nunes, F. Marques, F. Pereira, and A. Gasull, A contour-based approach to binary shape coding using multiple grid chain code, *Signal Process.: Image Commun.*, 15 (2000) 585–600.
34. K.J. O'Connell, Object-adaptive vertex-based shape coding method, *IEEE Trans. Circuits Syst.*, 7 (1997) 251–255.
35. J. Ostermann, Efficient encoding of binary shapes using MPEG-4, *Proceedings of IEEE International Conference on Image Processing, ICIP'98*, Chicago, USA, October 1998.
36. P. van Otterloo, *A Contour-Oriented Approach for Shape Analysis*, Prentice-Hall, Hertfordshire, UK, 1991.
37. T. Ozcelik and A.K. Katsaggelos, Very low bit rate video coding based on statistical spatio-temporal prediction of motion, segmentation and intensity fields, in *Video Data Compression for Multimedia Computing*, H.H. Li, S. Sun, and H. Derin, Eds., Kluwer, Norwell, MA, 1997, pp. 313–353.
38. R. Prasad, J.W. Vieveen, J.H. Bons, and J.C. Arnbak, Relative vector probabilities in differential chain coded linedrawings, *Proceedings of IEEE Pacific Rim Conference on Communication, Computers and Signal Processing*, Victoria, Canada, June 1989, pp. 138–142.
39. J.A. Saghri and H. Freeman, Analysis of the precision of generalized chain codes for the representation of planar curves, *IEEE Trans. Pattern Anal. Machine Intell.*, 3 (1981) 533–539.
40. P. Salama and C. Huang, Error concealment for shape coding, *Proceedings of IEEE International Conference on Image Processing*, vol. 2, pp. 701–704, Rochester, USA, Sept. 2002.
41. P. Salembier, F. Marques, and A. Gasull, Coding of partition sequences, in *Video Coding*, L. Torres and M. Kunt, Eds. Kluwer Academic Publishers, Englewood Cliffs, pp. 125–170, 1996.
42. G.M. Schuster and A.G. Katsaggelos, An optimal segmentation encoding scheme in the rate-distortion sense, *Proceedings of ISCAS 96*, Atlanta, USA, vol. 2, pp. 640–643.

43. S. Shirani, B. Erol, and F. Kossentini, A concealment method for shape information in MPEG-4 coded video sequences, *IEEE Trans. Multimedia*, 2 (2000) 185–190.
44. J. Signes, Binary format for scenes (BIFS): combining MPEG-4 media to build rich multimedia services, *Proceedings of the Visual Communications and Image Processing*, San Jose, CA, Jan. 1999.
45. L.D. Soares and F. Pereira, Refreshment need metrics for improved shape and texture object-based resilient video coding, *IEEE Trans. Image Process.*, 12 (2003) 328–340.
46. L.D. Soares and F. Pereira, Spatial shape error concealment for object-based image and video coding, *IEEE Trans. Image Process.*, 13 (2004) 586–599.
47. L.D. Soares and F. Pereira, Adaptive shape and texture intra refreshment schemes for improved error resilience in object-based video coding, *IEEE Trans. Image Process.*, 13 (2004) 662–676.
48. L.D. Soares and F. Pereira, Temporal shape error concealment by global motion compensation with local refinement, *IEEE Trans. Image Process.*, Vol. 15, No. 6, June 2006.
49. A. Vetro, T. Haga, K. Sumi, and H. Sun, Object-based coding for long-term archive of surveillance video, *Proceedings of IEEE International Conference on Multimedia and Expo, ICME'03*, Baltimore, USA, July 2003.
50. A. Vetro, H. Sun, and Y. Wang, MPEG-4 rate control for coding multiple video objects, *IEEE Trans. Circuits Syst. Video Technol.*, 9 (1999) 186–199.
51. A. Vetro, H. Sun, and Y. Wang, Object-based transcoding for adaptable video content delivery, *IEEE Trans. Circuits Syst. Video Technol.*, 11 (2001) 387–401.
52. A. Vetro, H. Sun, Y. Wang, and O. Gulyeruz, Rate-distortion modeling of binary shape using state partitioning, *Proceedings of the IEEE International Conference on Image Processing, ICIP'99*, Kobe, Japan, October 1999.
53. A. Vetro, Y. Wang, and H. Sun, Rate-distortion modeling for multiscale binary shape coding based on Markov Random Fields, *IEEE Trans. Image Process.*, 12 (2003) 356–364.
54. H. Wang, G.M. Schuster, A.K. Katsaggelos, and T.N. Pappas, An efficient rate-distortion optimal shape coding approach using a skeleton-based decomposition, *IEEE Trans. Image Process.*, 12 (2003) 1181–1193.
55. C.T. Zahn and R.Z. Roskies, Fourier descriptors for plane closed curves, *IEEE Trans. Comput.*, 21 (1972) 269–281.