Ostermann, Jörn “Object-Based Analysis–Synthesis Coding Based on Moving 3D Objects”
Multimedia Image and Video Processing
Ed. Ling Guan et al.
Boca Raton: CRC Press LLC, 2001
Chapter 10

Object-Based Analysis–Synthesis Coding Based on Moving 3D Objects

Jörn Ostermann

10.1 Introduction

For the coding of moving images with low data rates between 64 kbit/s and 2 Mbit/s, a block-based hybrid coder has been standardized by the ITU-T [8] where each image of a sequence is subdivided into independently moving blocks of size 16×16 picture elements (pels). Each block is coded by 2D motion-compensated prediction and transform coding [56]. This corresponds to a source model of “2D square blocks moving translationally in the image plane,” which fails at boundaries of naturally moving objects and causes coding artifacts known as blocking and mosquito effects at low data rates.

In order to avoid these coding distortions, several different approaches to video coding have been proposed in the literature. They can be categorized into four methods: region-based coding, object-based coding, knowledge-based coding, and semantic coding.

Region-based coding segments an image into regions of homogeneous texture or color [36]. Usually, these regions are not related to physical objects. Regions are allowed to move and change their shape and texture over time. Some recent proposals merge regions with dissimilar texture but with similar motion into one entity in order to increase coding efficiency [12].

The concept of object-based analysis–synthesis coding (OBASC) aiming at a data rate of 64 kbit/s and below was proposed in [44]. A coder based on this concept divides an image sequence into moving objects. An object is defined by its uniform motion and described by motion, shape, and color parameters, where color parameters denote luminance and chrominance reflectance of the object surface. Those parts of an image that can be described with sufficient accuracy by moving objects require the transmission of motion and shape parameters only, since the texture of the previously coded objects can be used. The remaining image areas are called areas of model failure. They require the transmission of shape and texture parameters in order to generate a subjectively correct decoded image. This detection of model failures can be adapted to accommodate properties of the human visual system. OBASC in its basic form does not require any a priori knowledge of the moving objects. A first implementation of an OBASC presented in 1991 was based on the source model of “moving flexible 2D objects” (F2D) and was used for coding image sequences between 64 and 16 kbit/s [3, 18, 26]. Implementations of an OBASC based on the source models of “moving rigid 3D objects” (R3D) and “moving flexible 3D objects” (F3D) using 3D motion and 3D shape are presented in [48] and [49], respectively. In [13], an implicit 3D shape representation is proposed.
Knowledge-based coders [31] use a source model which is adapted to a special object. In contrast to OBASC, this allows the encoding of only a special object like a face. However, due to this adaptation of the source model to the scene contents, a more efficient encoding becomes possible. A recognition algorithm is required to detect the object in the video sequence. For encoding of faces, a predefined 3D face model gets adapted to the face in the sequence. Then, the motion parameters of the face are estimated and coded. Perhaps the most challenging task for a knowledge-based encoder is the reliable detection of the face position [20, 34, 60].

Semantic coders [15] are modeled after knowledge-based coders. Until now, semantic coding was mainly investigated for the encoding of faces using high-level parameters such as the facial action coding system [2, 11, 14, 21, 22, 68]. Alternatively, facial animation parameters as defined by MPEG-4 can be used [30, 53]. By using high-level parameters we limit the degrees of freedom of the object and achieve a higher data reduction.

Knowledge-based as well as semantic coders use three-dimensional source models. Since they can encode only a particular object, they require a different algorithm for encoding the image areas outside this object. An OBASC based on a 3D source model seems to be a natural choice.

The purpose of this chapter is twofold. First, the concept of object-based analysis–synthesis coding is reviewed. Second, different source models used for OBASC and their main properties are compared. In order to use 3D source models, a reliable motion estimation algorithm is required. Here, we develop a robust gradient-based estimator that is able to track objects. The coding efficiencies obtained with the source models F2D [27], R3D, and F3D are compared in terms of data rate required for the same picture quality. The coding schemes will be evaluated using videophone test sequences. As a well-known reference for picture quality, the block-based hybrid coder H.261 [8, 9] is used.

In Section 10.2, the principles of object-based analysis–synthesis coding are reviewed. In Section 10.3, different source models used for OBASC and their implementations in an OBASC are presented. Since the data rate of an OBASC depends mainly on how well the image analysis can track the moving objects, we present in Section 10.4 details of image analysis. A robust 3D motion estimator is developed. Model failure detection is discussed in more detail.

In Section 10.5, we present an overview of parameter coding. The coding efficiency of the different source models is compared in Section 10.6. A final discussion concludes this book chapter.

### 10.2 Object-Based Analysis–Synthesis Coding

The goal of OBASC is the efficient encoding of image sequences. Each image of the sequence is called a real image. OBASC [44] subdivides each real image into moving objects called real objects. A real object is topologically connected and characterized by its uniform motion. A real object is modeled by a model object as defined by the source model of the encoder. Hence, one real object is described by one model object, whereas region-based coding describes regions with homogeneous textures as separate entities. Each model object \( m \) is described by three sets of parameters, \( A^{(m)} \), \( M^{(m)} \), and \( S^{(m)} \), defining its motion, shape, and color, respectively. Motion parameters define the position and motion of the object; shape parameters define its shape. Color parameters denote the luminance as well as the chrominance reflectance on the surface of the object. In computer graphics, they are sometimes called texture. The precise meaning of the three parameter sets depends on the source model employed (see Section 10.3).
Figure 10.1 is used to explain the concept and structure of OBASC. An OBASC consists of five parts: image analysis, parameter coding, parameter decoding, parameter memory, and image synthesis. Instead of the frame memory used in block-based hybrid coding, OBASC requires a memory for parameters in order to store the coded, transmitted, and decoded parameters $A'$, $M'$, and $S'$ for all objects. Whereas the double prime (′′) symbol marks the transmitted parameters used to update the parameter memory, the prime (′) symbol marks the decoded parameters at the output of the parameter memory.

**FIGURE 10.1**
Block diagram of an object-based analysis–synthesis coder.

The parameter memories in the coder and decoder contain the same information. Evaluating these parameter sets, image synthesis produces a model image $s_k'$, which is displayed at the decoder. In order to avoid annoying artifacts at object boundaries, a shape-dependent antialiasing filter may be applied at object boundaries [58].

At time instant $k + 1$, the image analysis has to evaluate the current image $s_{k+1}$ considering the parameter sets $A'$, $M'$, and $S'$ estimated for image $s_k$. The task of image analysis is to track each object known from previous frames and detect new moving objects. Each object $m$ is described by three sets of parameters, $A_{m,k+1}$, $M_{m,k+1}$, and $S_{m,k+1}$. These parameter sets are available at the output of the image analysis in PCM format. Considering the previously estimated and coded parameter sets $A'$, $M'$, and $S'$ creates a feedback loop in the encoder. This allows the image analysis to compensate for previous estimation errors as well as shape and motion quantization errors introduced by the lossy encoding of the parameters by parameter coding. Hence, an accumulation of estimation and quantization errors is avoided.

Figure 10.2 serves as an example to describe the parameter sets in the case of a source model of rigid 2D objects with 2D motion. The color parameters of an object can be covered and uncovered due to (1) camera motion, (2) a new object entering the scene, (3) motion of another object, or (4) egomotion. In this chapter, we focus on (2) to (4). The extension of OBASC to consider camera motion is straightforward on a conceptual level. Mech and Wollborn describe an implementation in [40]. In the example of Figure 10.2, areas of object 1 get uncovered due to the motion of object 2. Assuming that these parts of object 1 have not been visible before, the color parameters of these uncovered areas (UAs) have to be transmitted. Similarly, a rotating 3D object might uncover previously not visible areas for which the transmission of color parameters is required. The color parameters of an uncovered area can uniquely be associated with one object. Therefore, the color parameter $S_{m,k+1}$ of object $m$ at time instant $k + 1$ consists of its color parameter $S_{m,k}$ at time instant $k$ and the color parameters $S_{UA,m,k+1}$ of
FIGURE 10.2
Image analysis demonstrated using rigid translaterally moving objects in the image plane. The dashed lines denote the image boundaries in order to show the positions of object 2. It is not necessary to know the motion parameters at time instant $t_k$. The texture parameters of object 1 change due to the uncovered areas.

its uncovered area:

$$S_{m,k+1} = S_{m,k} \cup S_{m,k+1}^{UA}. \quad (10.1)$$

If several objects are moving, an uncovered area can belong to a moving object. The shape of the uncovered area can be derived from object motion and object shape. In Figure 10.2, the shape of the uncovered area of object 1 is determined by the shape and motion of object 2 [64].

Most implementations of an image analysis for an OBASC assume moving objects in front of a static background. In the current image, moving and static objects are detected first by means of change detection [23, 45, 52, 64]. For moving objects, new motion and shape parameters are estimated in order to reuse most of the already transmitted color parameters $S'(m)_k$. As pointed out in [25], the estimation of motion and shape parameters are mutually dependent problems. However, in the case of a static background, the correct estimation of motion is the more challenging task of the two [64]. Objects for which motion and shape parameters can be estimated successfully are referred to as MC objects (model compliance).

In the final step of image analysis, image areas that cannot be described by MC objects using the transmitted color parameters $S'(m)_k$ and the new motion and shape parameters $A_{k+1}^{(m)} M_{k+1}^{(m)}$, respectively, are detected. Areas of model failure (MF) [46] are derived from these areas. They are defined by 2D shape and color parameters only and are referred to as MF objects. The detection of MF objects takes into account that small position and shape errors of the MC objects — referred to as geometrical distortions — do not disturb subjective image quality. Therefore, MF objects are limited to those image areas with significant differences between the motion- and shape-compensated prediction image and the current image $s_{k+1}$. They tend to be small in size. This allows coding of color parameters of MF objects with high quality, thus avoiding subjectively annoying quantization errors. Since the transmission of color parameters
is expensive in terms of data rate, the total area of MF objects should not be larger than 4% of the image area, assuming 64 kbit/s, CIF (common intermediate format, 352×288 luminance and 176×144 chrominance pels/frame), and 10 Hz.

Depending on the object class MC/MF, the parameter sets of each object are coded by parameter coding using predictive coding techniques (Figure 10.3). Motion and shape parameters are encoded and transmitted for MC objects and shape and color parameters for MF objects. For MC objects, motion parameters are quantized and encoded. The motion information is used to predict the current shape of the MC object. After motion compensation of the shape, only the shape prediction error has to be encoded. The shape of uncovered areas is derived from the shape and motion of the MC objects. In the case of uncovered areas being visible for the first time, color parameters have to be transmitted. For MF objects, a temporal prediction of the shape is not useful, since areas of model failure are not temporally correlated. Hence, the shape parameters of MF objects are encoded in intra mode. For the color parameters, the motion-compensated prediction error is computed using the motion parameters of the underlying MC object. Then the prediction error is quantized and encoded. Table 10.1 summarizes which parameter sets have to be transmitted for each object class. As in any block-based coder, color parameters have to be transmitted only in the case of a scene cut. Since the coding of color parameters generally requires a bit rate of more than 1 bit/pel in active areas, the size of the MF objects determines to a large extent the bit rate required for encoding an image sequence. Hence, image analysis should be optimized such that the size of the MF objects becomes small. Furthermore, parameter coding for MF and MC objects has to be optimized in order to minimize the overall data rate $R = R_A + R_M + R_S$ for coding all parameter sets [50].

Parameter decoding decodes the two parameter sets transmitted for each object class. In the memory for parameters, the position and shape of MC objects are updated. Furthermore, in areas of model failure, color parameters of MC objects are substituted by the color parameters of the transmitted MF objects. Therefore, only MC objects are available at the output of the parameter memory.

In OBASC, the suitability of source models can be judged by comparing the data rates required for coding the same image sequence with the same image quality. Image quality is influenced mainly by the algorithm for detecting model failures and by the bit rate available.
for coding the color parameters of model failures. Assuming an image format of CIF with a reduced frame frequency of 10 Hz, an average area of MF objects of 4% of the image area should be sufficient in order to encode a videophone sequence with good subjective quality at a bit rate of 64 kbit/s.

### 10.3 Source Models for OBASC

In this section, the different source models applied to OBASC, their main properties, as well as some implementation details are presented. In order to highlight commonalities and differences between source models used for OBASC, it is useful to subdivide a source model into its main components, namely the camera model, the illumination model, the scene model, and the object model. The source model used here assumes a 3D real world that has to be modeled by a model world. Whereas the real image is taken by a real camera looking into the real world, a model image is synthesized using a model camera looking into the model world. A world is described by a scene, its illumination, and its camera. A scene consists of objects, their motion, and their relative position. Initially, the source models are distinguished from each other by the object model. For simplicity, we name the source models according to the name of their object model. Recent research also has focused on illumination models [62, 63].

The goal of the modeling is to generate a model world, $W_k$, with a model image identical to the real image, $s_k$, at a time instance $k$. This implies that the model objects may differ from the real objects. However, similarity between the real object and the model object generally helps in performing proper image analysis.

The following sections will describe the different parts of a source model. After the review of the camera, illumination, and scene model, different object models as applied to OBASC are explained. These object models are used to describe the real objects by means of MC objects. For each object model, parameter coding and some implementation details are highlighted.

#### 10.3.1 Camera Model

The real camera is modeled by a static pinhole camera [65]. Whereas a real image is generated by reading the target of the real camera, a model image is read off the target of the model camera. Assuming a world coordinate system $(x, y, z)$ and an image coordinate system $(X, Y)$, this camera projects the point $P^{(i)} = (P_x^{(i)}, P_y^{(i)}, P_z^{(i)})^T$ on the surface of an object in
the scene onto the point \( \mathbf{p}^{(i)} = (p_X^{(i)}, p_Y^{(i)})^T \) of the image plane according to

\[
p_X^{(i)} = F \cdot \frac{P_x^{(i)}}{P_z^{(i)}}, \quad p_Y^{(i)} = F \cdot \frac{P_y^{(i)}}{P_z^{(i)}}\tag{10.2}
\]

where \( F \) is the focal length of the camera (Figure 10.4). This model assumes that the image plane is parallel to the \((x, y)\) plane of the world coordinate system. For many applications, this camera model is of sufficient accuracy. However, in order to incorporate camera motion, a CAHV camera model [69] allowing for arbitrary camera motion and zoom should be used.

**FIGURE 10.4**
Camera model.

### 10.3.2 Scene Model

The scene model describes the objects of a world using an object model and the relationship between objects (Figures 10.2 and 10.5). It allows an explanation of the effects of covered and uncovered areas. In the case of uncovered areas, the relative position of the objects to each other and to the image plane allow the correct assignment of the area to one object.

### 10.3.3 Illumination Model

The illumination model describes the temporal changes in the video sequence caused by the changing illumination of the real world. The interaction of incident light from a light source with a point \( P \) of an object is described by the distribution \( L_{r,\lambda} \) of reflected radiance from an object surface depending on the distribution \( E_{i,\lambda} \) of incident irradiance and the object surface reflectance function \( R \) at this point according to

\[
L_{r,\lambda}(L, V, N, P, \lambda) = R(L, V, N, P, \lambda) \cdot E_{i,\lambda}(L, N, \lambda)\tag{10.3}
\]

Here \( N \) is the surface normal vector, \( L \) the illumination direction, \( V \) the viewing direction (to the focal point of the camera), and \( \lambda \) the wavelength of light (Figure 10.6). With simplifying
assumptions such as opaque object surfaces and temporally invariant illumination direction as well as viewing direction, (10.3) simplifies to

$$L_{r,\lambda}(N, P, \lambda) = R(N, P, \lambda) \cdot E_{i,\lambda}(N, \lambda)$$ (10.4)

Assuming the scene to be illuminated by a point light source and ambient diffuse light simplifies the description of the incident irradiance to the shading model of Phong used in early computer graphics [58]:

$$E_{i}(N) = c_{\text{ambient}} + c_{\text{lambert}} \cdot \max(0, LN)$$ (10.5)

with $c_{\text{ambient}}$ the ambient irradiance and $c_{\text{lambert}}$ the point light source irradiance. Assuming that $N$ and therefore the object shape is known, this simple illumination model requires three parameters to be estimated: the ratio between ambient and direct irradiance, $c_{\text{ambient}}/c_{\text{lambert}}$, and the two angles describing the direction of the direct point light source irradiation. This model according to (10.5) has been implemented in an OBASC by Stauder [62]. In the image
plane, Stander assumes that the luminance \( l(p) \) of a point moving from \( p_k \) to \( p_{k+1} \) changes according to

\[
l(p_{k+1}) = l(p_k) \cdot \frac{E_i(N_{k+1})}{E_i(N_k)}.
\]

(10.6)

Pearson and others proposed to model the irradiance by a discrete irradiance map [55]. This map gives an irradiance value for each patch of the incident light on a Gaussian sphere. It is not restricted to any illumination situation. Several light sources can be handled. For a reasonable approximation of an illumination situation, the number of patches should be \( 9 \times 9 \) or higher. This method is especially useful if the irradiance values can be measured.

Another simple illumination model assumes that the image signal \( l(p) \) depends on the illumination \( E(p) \) and the bidirectional reflection function \( R(p) \). \( R(p) \) accounts for the wavelength of the illumination, surface material, and the geometric arrangement of illumination, camera, and surface. The illumination \( E_i(p) \) depends on ambient and direct light. Assuming diffuse illumination, diffuse reflecting surfaces, parallel projection, and a constant \( k_b \), the image signal is given by the reflection model

\[
l(p) = k_B \cdot E_i(p) \cdot R(p).
\]

(10.7)

In the image plane, the luminance \( l(p) \) of a point moving from \( p_k \) to \( p_{k+1} \) changes according to

\[
l(p_{k+1}) = k_b \cdot E(p_{k+1}) \cdot R(p_k).
\]

(10.8)

This reflection model indicates that illumination can be modeled by a multiplicative factor. This has proven to be useful in block matching, 3D motion estimation, and change detection [7, 19, 52, 63].

The simplest, yet most widely used illumination model simply assumes for the luminance of a moving point

\[
l(p_{k+1}) = l(p_k).
\]

(10.9)

Sometimes, this model is referred to as the constant intensity assumption. The implicit assumptions are diffuse illumination, diffuse reflecting surfaces, and no temporal variation in the illumination. Here, we select the simple illumination model according to (10.9).

### 10.3.4 Object Model

The object model describes the assumptions of the source model about the real objects. In order to do so, shape, motion, and surface models are required. While all object models described here use the same surface model, they employ different motion and shape models as discussed below.

As far as the surface model is concerned, it is assumed that object surfaces are opaque and have a diffuse reflecting surface. The surface of an object \( m \) is described by the color parameters \( S_m \). These color parameters contain the luminance as well as the chrominance reflectance.

**Moving Rigid 2D Objects (R2D) with 3D Motion**

**Object Model**

This object model assumes rigid 2D arbitrarily shaped objects. Hence, each object can be perceived as a part of a plane. Its projection into the image plane is the 2D silhouette
of the object. Each object is allowed to move in 3D space. Allowing two parameters to describe the orientation of the plane in space and six parameters to describe object motion, the functional relationship between a point \( P \) on the object surface projected onto the image plane as \( p = (X, Y)^T \) and \( p' = (X', Y')^T \) before and after motion, respectively, is described by eight parameters \( (a_1, \ldots, a_8) \) [23, 65]:

\[
p' = (X', Y')^T = \left( \frac{a_1X + a_2Y + a_3}{a_7X + a_8 + 1}, \frac{a_4X + a_5Y + a_6}{a_7X + a_8 + 1} \right) \cdot (10.10)
\]

**Implementation**

Image analysis for this source model estimates motion hierarchically for an object, which initially is the entire frame, \( s_{k+1} \). Then, the motion-compensated prediction \( \hat{s} \) of the object is computed using an image synthesis algorithm, which fetches the luminance for a point \( p' \) in frame \( s_{k+1} \) from point \( p \) of frame \( s_k \) according to the inverse of (10.10). Finally, the estimated motion parameters are verified by comparing the original image and the predicted image, and detecting those areas where the motion parameters do not allow for a sufficiently precise approximation of the original image. These areas are the objects where motion parameters are estimated in the next step of the hierarchy. This verification step allows the segmentation of moving objects using the motion as the segmentation criterion. Hötter [23] implemented three steps of the hierarchy. The image areas that could not be described by motion parameters at the end of this object segmentation and motion compensation process are the MFR2D objects.

In order to achieve robust estimates, Hötter allowed the motion model to reduce the number of parameters from eight to an affine transformation with six \( (a_1, \ldots, a_6) \) parameters or to a displacement with two parameters \( (a_1, a_2) \). This adaptation is especially important as objects become smaller [23].

In order to increase the efficiency of parameter coding, especially the shape parameter coding, the segmentation of the current frame into moving objects takes the segmentation of the previous image as a starting point. In the image plane, this increases the temporal consistency of the 2D shape of the objects. Since eight parameters are estimated for each frame in which an object is visible, this model allows the object to change its orientation in space arbitrarily from one frame interval to the next. No temporal coherence for the orientation of the object is required or enforced.

**Moving Flexible 2D Objects (F2D) with 2D Motion**

**Object Model**

This source model assumes that the motion of a real object can be described by a homogeneous displacement vector field. This displacement vector field moves the projection of the real object into the image plane to its new position. Assuming a point \( P \) on the object surface moving from \( P \) to \( P' \), its projection into the image plane moves from \( p \) to \( p' \). \( p \) and \( p' \) are related by the displacement vector \( \tilde{D}(p') = (D_X(p'), D_Y(p'))^T \):

\[
p = p' - \tilde{D}(p') \cdot (10.11)
\]

The motion parameters of an MC object \( m \) are the displacement vectors of those points \( p' \) that belong to the projection of object \( m \) into the image plane. The shape of this object is defined by the 2D silhouette that outlines the projection of object \( m \) in the image plane.

**Implementation**

For estimating the changed area due to object motion, Hötter applies a change detector to the coded image \( s_k' \) and the real current image \( s_{k+1} \). This change detector is initialized with the
silhouette of the objects as estimated for image $s_k$. This allows object tracking and increases the coding efficiency for shape parameters. For displacement estimation, a hierarchical block-matching technique is used [4]. Experimental investigations show that an amplitude resolution of a half pel and a spatial resolution of one displacement vector for every $16 \times 16$ pel result in the lowest overall data rate for encoding. This implementation of image analysis is only able to segment moving objects in front of a static background. Since an OBASC relies on the precise segmentation of moving objects and their motion boundaries, this restriction on the image analysis limits the coding efficiency for scenes with complex motion.

To compute the motion-compensated prediction image, the texture of the previously decoded image can be used similar to the MPEG-1 and MPEG-2 standards. The displacement vector field is bilinearly interpolated inside an object. The vectors are quantized to half-pel accuracy. In order to accomplish prediction using these half-pel motion vectors, the image signal is bilinearly interpolated. In [27], the disadvantage of this image synthesis by filter concatenation was noted. Assuming that all temporal image changes are due to motion, the real image $s_k(p')$ is identical to $s_0(p)$. If displacement vectors with subpel amplitude resolution are applied, the displaced position does not necessarily coincide with the sampling grid of $s_0$. In that case, a spatial interpolation filter $h$ is required to compute the missing sample. In Figure 10.7, the luminance value of $s_{k+1}(y_1)$ requires two temporal filter operations. Assuming bilinear interpolation and the displacement vector field as depicted in Figure 10.7, $s_{k+1}(y_1)$ is given by

$$s_{k+1}(y_1) = s_k(y) * h(y - D_k(y))$$

$$= s_0(y) * h(y - D_{k+1}(y)) * h(y - D_k(y))$$

FIGURE 10.7
Image synthesis by filter concatenation (one-dimensional case) [27].

The disadvantage of this method is that repeated interpolation results in severe low-pass filtering of the image. Hötter suggests image synthesis by parameter concatenation using an object memory for the texture parameters of each object (Figure 10.8). Thus, the interpolation filter $h$ has to be applied only once. In order to synthesize the image $s_{k+1}(y_1)$, the total displacement

\[
s_{k+1} (y_1) = \frac{1}{4} s_0 (y_1) + \frac{1}{2} s_0 (y_2) + \frac{1}{4} s_0 (y_3) .
\]
between $s_0$ and $s_{k+1}$ is computed by concatenating the displacement vectors. In this case, $s_{k+1}(y_1)$ is given by

$$s_{k+1}(y_1) = s_0(y_2) .$$

(10.13)

This is basically a method of texture mapping, as known from computer graphics, and OBASC based on 3D source models (see Section 10.3.4) [67]. Indeed, Hötter’s implementation uses a mesh of triangles in order to realize this object memory, or texture memory as it is known in computer graphics (Figure 10.9). In [28], Hötter develops a stochastic model describing the synthesis errors due to spatial interpolation and displacement estimation errors. The model was verified by experiments. Use of this texture memory gives a gain of 1 dB in the signal-to-noise ratio for every 14 frames encoded. This gain is especially relevant since the improvement is only due to the less frequent use of the interpolation filter, thus resulting in significantly sharper images.

FIGURE 10.8
Image synthesis by parameter concatenation (one-dimensional case) using an object memory for color parameters [27].
FIGURE 10.9
Image synthesis for MC objects using a triangle-based mesh as texture memory.

MF objects are detected as described in Section 10.4.3. The shape parameter coding is presented in Section 10.5.2. The 2D displacement vector fields are DPCM coded using spatial prediction.

Moving Rigid 3D Objects (R3D)

Object Model
In the model used here, the 3D shape is represented by a mesh of triangles, which is put up by vertices referred to as control points, $P(i)$. The appearance of the model object surface is described by the color parameters $S(m)$. In order to limit the bit rate for coding of shape parameters, the shape parameters $M^{(m)}$ of an object $m$ represent a 2D binary mask, which defines the silhouette of the model object in the model image. During initialization, the 3D shape of an object is completely described by its 2D silhouette (i.e., there is an algorithm that computes a generalized 3D cylinder from a 2D silhouette (Figure 10.10) using a distance transform to determine the object depth (Figure 10.10b) [45]). The distance transform assigns two depth values $w_{F \pm Z}$ to each point $h$ of the object silhouette. Each depth value depends on the Euclidian distance between point $h$ and the silhouette boundary. Depending on the application, an appropriate mapping $d \rightarrow w_{F \pm Z}$ can be selected. Using a mapping derived from an ellipse is suitable for the modeling of head and shoulder scenes. The object width $b$ and the object depth $h$ are related according to (Figure 10.11):

$$w_{F \pm Z}(d) = F \pm \begin{cases} \frac{h}{b} \sqrt{d(b - d)} & \text{for } d < \frac{b}{2} \\ \frac{h}{2} & \text{otherwise} \end{cases}$$

(10.14)

In order to determine the object width, we use the four measurements $b_1, b_2, b_3,$ and $b_4$ according to Figure 10.12 in order to determine the maximum elongation of the object in the direction of the $x$ and $y$ axes as well as the two diagonals. We determine the object width $b$ as
FIGURE 10.10
Processing steps from object silhouette to model object: (a) object silhouette; (b) 3D object shape with required silhouette rotated by 30° and illuminated; (c) contour lines approximating the object shape; (d) polygons approximating the contour lines; (e) mesh of triangles using polygon points as vertices; (f) model object with color parameters projected onto it.

the minimum of the four values according to

\[ b = \min(b_1, b_2, b_3, b_4). \]  

(10.15)

\( w_{F \pm Z} \) is constant for \( d > b/2 \). Hence, the surface of the model object is parallel to the image plane where the object is wide. In order to automatically adapt the object depth to the width of different objects, we set the ratio \( \beta = b/h \) instead of \( h \) to a fixed value:

\[
w_{F \pm Z}(d) = F \pm \begin{cases} 
\frac{1}{\beta} \sqrt{d(b - d)} & \text{for } d < \frac{b}{2} \\
\frac{b}{\beta} & \text{otherwise}.
\end{cases}
\]

(10.16)

In Figure 10.10, the ratio \( \beta \) between object width and object depth is set to 1.5. The maximum distance between the estimated silhouette and the silhouette of the model object does not exceed \( d_{\text{max}} \leq 1.4 \text{ pel} \) (Figure 10.10d). After initialization, the shape parameters \( M^{(m)} \) are used as update parameters to the model object shape.

An object may consist of one, two, or more rigid components [6]. The subdivision of an object into components is estimated by image analysis. Each component has its own set of motion parameters. Since each component is defined by its control points, the components are linked by those triangles of the object having control points belonging to different components. Due to these triangles, components are flexibly connected. Figure 10.13 shows a scene with the objects Background and Claire. The model object Claire consists of the two components Head and Shoulder.
FIGURE 10.11
A 3D shape symmetric to the image plane \( Z = F \) is created. (a) Distance transform according to (10.14); \( d \) is the smallest distance to the border of the object silhouette, \( b \) is set according to (10.15), and \( \beta = b/h = 1.5 \). In the example, the object width is larger than \( b \) according to 10.15. (b) Cut through a model object (top); view from the focal point of the camera onto the contour lines (bottom). For computing object depth, we always measure the distance \( d \) to the closest boundary point [51].

FIGURE 10.12
Determining the object width. The distances \( b_1, b_2, b_3, \) and \( b_4 \) are measured by determining the parallel projection of the silhouette onto the \( x \) and \( y \) axes and onto the image diagonals. We use the minimum as object width, here \( b_1 \).

3D motion is described by the parameters \( A^{(m)} = (T_{x}^{(m)}, T_{y}^{(m)}, T_{z}^{(m)}, R_{x}^{(m)}, R_{y}^{(m)}, R_{z}^{(m)}) \) defining translation and rotation. A point \( P^{(i)} \) on the surface of object \( m \) with \( N \) control points \( P^{(i)} \) is moved to its new position \( P^{(i)}' \) according to

\[
P^{(i)}' = R^{(m)} \cdot (P^{(i)} - C^{(m)}) + C^{(m)} + T^{(m)}
\]

with the translation vector \( T^{(m)} = (T_{x}^{(m)}, T_{y}^{(m)}, T_{z}^{(m)})^T \), the object center \( C = (C_x, C_y, C_z) = \)
FIGURE 10.13
Model scene and model object Claire subdivided into two flexibly connected components: (a) scene consisting of two objects; (b) components of model object Claire.

FIGURE 10.14
Triangular mesh with color parameter on the skin of the model object.

FIGURE 10.15
Triangular mesh after flexible shape compensation by flexible shift vector $F^{(n)}$. 
\[ \frac{1}{N} \sum_{i=1}^{N} P_C^{(i)} \] the rotation angles \( R_C = (R_x^{(C)}, R_y^{(C)}, R_z^{(C)})^T \), and the rotation matrix \([R_C]\) defining the rotation in the mathematically positive direction around the \( x, y, \) and \( z \) axes with the rotation center \( C \):

\[
[R_C] = \begin{bmatrix}
\cos R_y \cos R_z & \sin R_x \sin R_z - \cos R_x \cos R_z & \sin R_x \sin R_z + \cos R_x \cos R_z \\
-\sin R_y & \cos R_y \cos R_z & \sin R_y \sin R_z \\
\cos R_x \sin R_z - \sin R_x \cos R_z & \cos R_x \sin R_z + \cos R_x \cos R_z & \cos R_x \cos R_z 
\end{bmatrix}
\]

(10.18)

**Implementation**

An overview of the image analysis and a detailed description of motion estimation is given in Section 10.4. Parameter coding is presented in Section 10.5.

**Moving Flexible 3D Objects (F3D)**

**Object Model**

In addition to the properties of the source model R3D, the source model F3D allows for local flexible shifts on the surface of the model object shape. This is modeled by a flexible skin (Figure 10.14). This flexible skin can be moved tangentially to the surface of the object (Figure 10.15). It allows modeling of local deformations. In the model world, the flexible surface is modeled by a shift of control points \( P_C^{(n)} \) in the tangential surface plane. The normal vector to this tangential surface plane is computed by averaging the normal vectors \( n_{Dj}^{(n)} \) to the \( J \) triangles to which the control point \( P_C^{(n)} \) belongs:

\[
n_{t}^{(n)} = \sum_{j=1}^{J} n_{Dj}^{(n)}
\]

(10.19)

This tangential surface plane with the normal vector \( n_{t}^{(n)} \) is spanned by \( R_{fu}^{(n)} \) and \( R_{fv}^{(n)} \). These vectors are of unit length and are orthogonal to each other. For each control point \( P_C^{(n)} \), two flexible shape parameters \( S_{fu}^{(n)} = (S_{fu}^{(n)}, S_{fv}^{(n)})^T \) have to be estimated.

\[
P_C^{(n)} = P_C^{(n)} + F^{(n)}
\]

\[
P_C^{(n)} = P_C^{(n)} + S_{fu}^{(n)} R_{fu} + S_{fv}^{(n)} R_{fv}
\]

(10.20)

with \( F \) the flexible shift vector and \( P_C^{(n)} = (P_x, P_y, P_z)^T \) and \( P_C^{(n)} = (P'_x, P'_y, P'_z)^T \) being a control point before and after shift, respectively. The flexible shift vectors \( F^{(n)} \) can be interpreted as local motion parameters, in contrast to the global motion parameters, \( R^{(m)} \) and \( T^{(m)} \).

**Implementation**

The flexible shift vectors require additional data rates for encoding. Hence they are estimated and transmitted only for those image areas that cannot be described with sufficient accuracy using the source model R3D (see Section 10.4.1) [49]. These areas are the MF_{R3D} objects. The shift parameters are estimated for one MF_{R3D} object at a time. For all control points of an MC object that are projected onto one MF_{R3D} object, the shift parameters are estimated jointly because one control point affects the image synthesis of all the triangles to which it belongs. For estimation, the image signal is approximated by a Taylor series expansion and the parameters are estimated using a gradient method similar to the 3D motion estimation.
algorithm described in Section 10.4.2. Since the robust 3D motion estimation as described in Section 10.4.2 is not influenced by model failures, it is not necessary to estimate the 3D motion parameters again or to estimate them jointly with the flexible shift parameters.

10.4 Image Analysis for 3D Object Models

The goal of image analysis is to gain a compact description of the current real image $s_{k+1}$, taking the transmitted parameter sets $A_{k+1}^{(m)}$, $M_{k+1}^{(m)}$, and $S_{k+1}^{(m)}$ for each object $m$. First, a model image, $s_{k}'$, of the current model world is computed by means of image synthesis. In order to compute the change detection mask, $B_{k+1}$, the change detection evaluates the images $s_{k}'$ and $s_{k+1}$ on the hypothesis that moving real objects generate significant temporal changes in the images [23, 64], that they have occluding contours [45], and that they are opaque [45, 63]. This mask $B_{k+1}$ marks the projections of moving objects and the background uncovered due to object motion as changed. Areas of moving shadows or illumination changes are not marked as changed because illumination changes can be modeled by semitransparent objects [62].

Since change detection accounts for the silhouettes of the model objects, the changed areas in mask $B_{k+1}$ will be at least as large as these silhouettes. The inputs to image analysis are the current real image, $s_{k+1}$, and the model world, $W'$, described by its parameters $A_{k+1}^{(m)}$, $M_{k+1}^{(m)}$, and $S_{k+1}^{(m)}$ for each object $m$. First, a model image, $s_{k}'$, of the current model world is computed by means of image synthesis.

In order to compute the change detection mask, $B_{k+1}$, the change detection evaluates the images $s_{k}'$ and $s_{k+1}$ on the hypothesis that moving real objects generate significant temporal changes in the images [23, 64], that they have occluding contours [45], and that they are opaque [45, 63]. This mask $B_{k+1}$ marks the projections of moving objects and the background uncovered due to object motion as changed. Areas of moving shadows or illumination changes are not marked as changed because illumination changes can be modeled by semitransparent objects [62].

Since change detection accounts for the silhouettes of the model objects, the changed areas in mask $B_{k+1}$ will be at least as large as these silhouettes. The applied motion estimation algorithm requires motion, shape, and color parameters of the model objects and the current real image $s_{k+1}$ as input. Motion parameters $A_{k+1}$ are estimated using a Taylor series expansion of the image signal, linearizing the rotation matrix (10.18) assuming small rotation angles and maximum likelihood estimation (see Section 10.4.2) [29].

The resulting motion parameters are used to detect the uncovered background, which is included in mask $B_{k+1}$. The basic idea for the detection of uncovered background is that the projection of the moving object before and after motion has to lie completely in the changed area [23]. Subtracting the uncovered background from mask $B_{k+1}$ gives the new silhouette $C_{k+1}$ for all model objects (Figure 10.17).

The silhouette of each model object $m$ is then compared and adapted to the real silhouette $C_{k+1}^{(m)}$ [45]. Differences occur either when parts of the real object start moving for the first time or when differences between the shape of the real and the model object become visible.
FIGURE 10.16
Block diagram of image analysis: \(A'_k, M'_k, S'_k\) stored motion, shape, and color parameters; \(s_{k+1}\) real image to be analyzed; \(s'_k, s^*\) model images; \(B_{k+1}\) change detection mask; \(C_{k+1}\) object silhouettes; \(M_{k+1}\) shape parameters for MC and MF objects; \(S_{k+1}\) color parameters for MC and MF objects. Arrows indicate the information used in various parts of image analysis. Semicircles indicate the output of processing steps.

During rotation, in order to compensate for the differences between the silhouettes of the model objects and \(C_{k+1}\), the control points close to the silhouette boundary are shifted perpendicular to the model object surface such that the model object gets the required silhouette. This gives the new shape parameters \(M_{k+1}^{MC}\), where MC denotes model compliance.

For the detection of model failures, a model image \(s^*\) is synthesized using the previous color parameters \(S'_k\) and the current motion and shape parameters \(A_{k+1}\) and \(M_{k+1}^{MC}\), respectively. The differences between the images \(s^*\) and \(s_{k+1}\) are evaluated for determining the areas of model failure. The areas of model failure cannot be compensated for using the source model of “moving rigid 3D objects.” Therefore, they are named rigid model failures (MF_{R3D}) and are represented by MF_{R3D} objects. These MF objects are described by 2D shape parameters \(M_{k+1}^{MF}\) and color parameters \(S_{k+1}^{MF}\) only.

In case the source model F3D is used, three more steps have to be added to the image analysis (Figure 10.18). As explained in Section 10.3.4, flexible shift parameters are estimated only for those parts of the model object that are projected onto areas of MF_{R3D}. Following the example of Figure 10.23, the estimation is limited to control points that are projected onto the eye, mouth, and right ear area. After estimation of the shift parameters, a model image is
FIGURE 10.17
Object silhouettes of Claire: (a) frame 2; (b) frame 18. Gray and black areas are marked changed by change detection. Detection of object silhouettes gives the object silhouette (gray) and the uncovered background (black).

synthesized and the areas of MF_{3D} are estimated using the same algorithm for the detection of model failures as for the R3D source model.

10.4.2 Motion Estimation for R3D

Image analysis of the OBASC must compensate for the motion of the objects in the scene in order to provide a motion-compensated prediction of the current image s_{k+1}. The 3D real objects are modeled by 3D model objects. Usually, the motion and the shape of the real objects are unknown. In order to ensure a reliable and robust estimation, methods for robust estimation are used. In Section 10.4.2, the basic motion estimation algorithm, which enables tracking of real objects with model objects, is reviewed [48]. In Section 10.4.2, methods for robust estimation are developed and compared. Since the shape of the real objects is not known and these objects tend not to be completely rigid due to facial expressions and hair motion, we developed a gradient-based motion estimator instead of a feature-based estimator.

Basic Motion Estimation

In order to derive the motion estimation algorithm, it is assumed that differences between two consecutive images s_k and s_{k+1} are due to object motion only. In order to estimate these motion parameters, a gradient method is applied here.

During motion estimation, each object is represented by a set of observation points. Each observation point O^{(j)} = (Q^{(j)}, g^{(j)}, I^{(j)}) is located on the model object surface at position Q^{(j)} and holds its luminance value I^{(j)} and its linear gradients \( g^{(j)} = (g_x^{(j)}, g_y^{(j)})^T \). \( g_Q \) are the horizontal and vertical luminance gradients from the image which provided the color parameters for the object. For simplicity, we use \( s_k \) here. The gradients are computed by convoluting the image signal with the Sobel operator

\[
E = \frac{1}{8} \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}
\]

(10.21)

giving the gradients

\[
g_x(x, y) = l(x, y)^T E
\]
\[
g_y(x, y) = l(x, y)^T E^T.
\]

(10.22)
FIGURE 10.18
Block diagram of image analysis: $A_k'$, $M_k'$, and $S_k'$ stored motion, shape, and color parameters; $s_{k+1}$ real image to be analyzed; $s^*$ model image; $A_{k+1} = A_{MC}^{R3D} + A_{MC}^{F3D}$ global $(R, T)$ and local $(S_f)$ motion parameters of MC objects; $M_{k+1} = M_{MC}^{R3D} + M_{MF}^{F3D}$ shape parameters of MC and MF objects; $S_{k+1} = S_k' + S_{k+1}^{MF}$ color parameters of MC and MF objects. Arrows indicate the information used in various parts of image analysis. Semicircles indicate the output of processing steps.

The measure for selecting observation points is a high spatial gradient. This adds robustness against noise to the estimation algorithm (see Section 10.4.2). Figure 10.19 shows the location of all observation points belonging to the model object Claire. If parts of the object texture are exchanged due to MF objects, the observation points for the corresponding surface of the object are updated. The observation points are also used for the estimation of flexible shift parameters.

FIGURE 10.19
The position of all observation points of model object Claire.
Since some triangles of a model object can be deformed due to flexible shifts or due to its control points belonging to different components of the model object, we define the position of an observation point \( Q(j) \) relative to the position of the control points \( P(0), P(1), \) and \( P(2) \) of its triangle using barycentric coordinates \( c_0, c_1, \) and \( c_2 \) of the coordinate system defined by \( P(0), P(1), \) and \( P(2): \)

\[
Q(j) = c_0 P(0) + c_1 P(1) + c_2 P(2).
\]

(10.23)

It is assumed that objects are rigid and have diffuse reflecting surfaces. Furthermore, diffuse illumination of the scene is assumed. Hence, color parameters are constant. With an observation point \( O(j)_k \) at time instant \( k \) projected onto the image plane at \( q(j)_k \) and the same observation point after motion \( O(j)_{k+1} = (Q(j)_{k+1}, g(j), I(j)) \) projected onto \( q(j)_{k+1} \), the luminance difference between image \( k \) and image \( k + 1 \) at position \( q(j)_k \) is

\[
\Delta I(q(j)_{k}) = s_{k+1}(q(j)_{k+1}) - s_k(q(j)_{k}) \approx s_{k+1}(q(j)_{k}) - s_k(q(j)_{k}) = s_{k+1}(q(j)_{k}) - s_k(q(j)_{k}) + \bar{g} \cdot (q(j)_{k+1} - q(j)_k),
\]

(10.24)

assuming that the luminance difference is only due to object motion with \( s_{k+1}(p_{k+1}) = s_k(p_k) = I \). According to [4], we can approximate the image signal using a Taylor expansion of second order without explicitly computing the second derivative. We compute the second-order gradient \( \bar{g} \) by averaging the linear gradients of the observation point and the image signal

\[
\bar{g} = \frac{1}{2} \left( g_Q + g_{k+1}(q) \right).
\]

(10.25)

Approximating the image signal with a Taylor series and stopping after the linear term gives

\[
s_{k+1}(q_{k+1}) \approx s_{k+1}(q_k) + \bar{g} \cdot (q_{k+1} - q_k).
\]

(10.26)

Now, we can express the luminance difference according to (10.24) as

\[
\Delta I(q(j)_{k}) = -\bar{g} \cdot (q(j)_{k+1} - q_k) = \left( g_{x(j)}^{(j)} \right)^T \cdot \left( q(j)_{k+1} - q(j)_k \right).
\]

(10.27)

Substituting image coordinates by model world coordinates with equation (10.2) yields

\[
\Delta I(j) = F \cdot g_{x(j)} \left( \frac{Q(j)_{k+1}}{Q(j)_{k+1}} - \frac{Q(j)_{k+1}}{Q(j)_{k+1}} \right) + F \cdot g_{y(j)} \left( \frac{Q(j)_{k+1}}{Q(j)_{k+1}} - \frac{Q(j)_{k+1}}{Q(j)_{k+1}} \right)
\]

(10.28)

The position \( Q(j) \) of the observation point \( O(j) \) is known. By relating \( Q(j) \) to \( Q(j+1) \) by means of the motion equation (10.17), a nonlinear equation with the known parameters \( \Delta I, g, \) and \( F \) and the six unknown motion parameters results. This equation is linearized by linearizing the rotation matrix \( R_C \) (10.18), assuming small rotation angles

\[
[R_C'] = \begin{bmatrix}
1 & -R_z & R_y \\
R_z & 1 & -R_x \\
-R_y & R_x & 1
\end{bmatrix}
\]

(10.29)

\[1\text{See [7] and [62] on how to consider illumination effects.}\]
giving

\[ Q_{k+1} = \left[R'_C \right] \cdot (Q_k - C) + C + T \]  \hspace{1cm} (10.30)

Substituting (10.30) into (10.28), the linearized equation for one observation point is

\[
\begin{align*}
\Delta f &= F \cdot g_x/Q_z \cdot T_x \\
&+ F \cdot g_y/Q_z \cdot T_y \\
&- \left[ (Q_x g_x + Q_y g_y) F/Q_z^2 + \Delta I/Q_z \right] \cdot T_z \\
&- \left[ (Q_x g_x (Q_y - C_y) + Q_y g_y (Q_y - C_y) + Q_z g_x (Q_z - C_z)) F/Q_z^2 \\
&+ \Delta I/Q_z (Q_y - C_y) \right] \cdot R_x \\
&+ \left[ (Q_y g_y (Q_x - C_x) + Q_x g_x (Q_x - C_x) + Q_z g_x (Q_z - C_z)) F/Q_z^2 \\
&+ \Delta I/Q_z (Q_x - C_x) \right] \cdot R_y \\
&- \left[ g_x (Q_y - C_y) - g_y (Q_x - C_x) \right] F/Q_z \cdot R_z \\
\end{align*}
\]  \hspace{1cm} (10.31)

with the unknown motion parameters \( T = (T_x, T_y, T_z)^T \) and \( R_C = (R_x, R_y, R_z)^T \) and the observation point \( O_k = (Q_k, g, I) \) at position \( Q_k = (Q_x, Q_y, Q_z)^T \). In order to get reliable estimates for the six motion parameters, equation (10.31) has to be established for many observation points, resulting in an overdetermined system of linear equations

\[ A \cdot x - b = r, \]  \hspace{1cm} (10.32)

with the residual \( r = (r_1, \ldots, r_J)^T \), \( x = (T_x, T_y, T_z, R_x, R_y, R_z)^T \), \( b = (\Delta I(q^{(1)}), \ldots, \Delta I(q^{(J)})^T \), and \( A = (a_1, \ldots, a_J)^T \), and \( a_j \) according to (10.31). The equations are solved by minimization of \( r \):

\[ |r|^2 = r^T \cdot r \rightarrow \min, \]  \hspace{1cm} (10.33)

which corresponds to a minimization of the prediction error of the observation points

\[ \sum_{0\leq j \leq J} (\Delta I^{(j)})^2 \rightarrow \min. \]  \hspace{1cm} (10.34)

The motion parameters are given by

\[ \hat{x} = \left( A^T \cdot A \right)^{-1} \cdot A^T \cdot b \]  \hspace{1cm} (10.35)

In order to avoid the inversion of large matrices, we do not compute \( A \) but immediately compute the 6 \( \times \) 6 matrix \( A^T \cdot A \).

Due to the linearizations in (10.26) and (10.29), motion parameters have to be estimated iteratively for each model object. After every iteration, the model object is moved according to (10.17) using the estimated motion parameters \( \hat{x} \). Then, a new set of motion equations is established, giving new motion parameter updates. Since the motion parameter updates approach zero during the iterations, the introduced linearizations do not harm motion estimation. The iteration process terminates if the decrease of the residual error \( |r|^2 \) becomes negligible.
Robust Motion Estimation

Equation (10.32) is solved such that the variance of the residual errors $\Delta I$ is minimized. However, this approach is sensitive to measurement errors [41]. Measurement errors occur because (10.32) is based on several model assumptions and approximations that tend to be valid for the majority of observation points but not all. Observation points that violate these assumptions are named outliers [59]. When using (10.34) for solving (10.32), outliers have a significant influence on the solution. Therefore, we have to take measures that limit the influence of these outliers on the estimation process [51]. Sometimes, the following assumptions are not valid:

1. Rigid real object
2. Quadratic image signal model
3. Small deviations of model object shape from real object shape

Each of these cases is discussed below.

If parts of the real object are nonrigid (i.e., the object is flexible), we have image areas that cannot be described by the current motion and shape parameters $A_i$ and $M_i$, respectively, and the already transmitted color parameters $S_i'$. These image areas can be detected due to their potentially high prediction error $\Delta I$. Observation points in these areas can be classified as outliers. For iteration $i$ of (10.34), we will consider only observation points for which the following holds true:

$$\Delta I_{i}^{(j)} < \sigma_{\Delta I} \cdot T_{ST}$$  \hspace{1cm} (10.36)

with

$$\sigma_{\Delta I} = \sqrt{\frac{1}{J} \sum_{j=0}^{J} (\Delta I_{i}^{(j)})^2}. \hspace{1cm} (10.37)$$

The threshold $T_{ST}$ is used to remove the outliers from consideration.

According to (10.26), motion estimation is based on the gradient method, which allows for estimating only small local displacements $(q_{k+1}^{(j)} - q_{i}^{(j)})$ in one iteration step [43]. Given an image gradient $g_{i}^{(j)}$ and a maximum allowable displacement $V_{max} = |v_{max}| = |(v_{x,max}, v_{y,max})^T|$, we can compute a maximum allowable frame difference $\Delta I_{limit}(q_{i}^{(j)})$ at an image location $q_{i}^{(j)}$

$$\Delta I_{limit}(q_{i}^{(j)}) = |v_{max} \cdot g_{i}^{(j)}|. \hspace{1cm} (10.38)$$

Observation points with $|\Delta I(q_{i}^{(j)})| > |\Delta I_{limit}(q_{i}^{(j)})|$ are excluded from consideration for the $i$th iteration step. We assume that they do not conform to the image signal model assumption.

Considering image noise, we can derive an additional criterion for selecting observation points. Assuming white additive camera noise $n$, we measure the noise of the image difference signal as

$$\sigma_{\Delta I}^2 = 2 \cdot \sigma_n^2. \hspace{1cm} (10.39)$$

According to (10.27), we represent the local displacement $(q_{k+1}^{(j)} - q_{i}^{(j)})$ as a function of the noiseless luminance signal. Therefore, the luminance difference $\Delta I(q_{i}^{(j)})$ and the gradient
$g^{(j)}$ should have large absolute values in order to limit the influence of camera noise. Hence, we select as observation points only points with a gradient larger than a threshold $T_G$:

$$\left| g^{(j)} \right| > T_G.$$  \hfill (10.40)

Relatively large gradients allow also for a precise estimation of the motion parameters. Summarizing these observations, we conclude that we should select observation points with large absolute image gradients according to (10.40). Equations (10.36) and (10.38) are the selection criteria for the observation points we will use for any given iteration step.

Instead of using the binary selection criteria for observation points according to (10.36) and (10.38), we can use continuous cost functions to control the influence of an observation point on the parameter estimation. We use the residuum $r$ according to (10.32) as measure for the influence of an observation point [57, 70]. Assuming that the probability density function $f(r)$ of the residuals $r_j$ according to (10.32) is Gaussian, (10.34) is a maximum-likelihood estimator or M estimator [29].

Now, we will investigate how different assumptions about $f(r)$ influence the M estimator. Let us assume that one $f(r)$ is valid for all observation points. A critical point for selecting an appropriate probability density function is the treatment of outliers. Ideally, we want outliers to have no influence on the estimated motion parameters.

The M estimator minimizes the residuum $r_j$ according to (10.32) using a cost function $\varrho(r_j)$:

$$J \sum_{j=1}^{J} \varrho(r_j) \rightarrow \min_x.$$  \hfill (10.41)

With

$$\Psi(r_j) = \frac{\delta(\varrho(r_j))}{\delta x}$$  \hfill (10.42)

the solution of (10.41) becomes

$$J \sum_{j=1}^{J} \Psi(r_j) = 0.$$  \hfill (10.43)

Equation (10.43) becomes an M estimator for the probability density function $f(r)$ if we set

$$\varrho(r) = -\log f(r).$$  \hfill (10.44)

This M estimator is able to compute the correct solution for six motion parameters with up to 14% of the observation points being outliers [70]. Some authors report success with up to 50% outliers [38].

Let us assume that $\varepsilon\%$ of our measurement data represents outliers that do not depend on the observable motion. Now, we can choose a separate probability density function for the residuals of the outliers. Let us assume that the residuals of the inliers (non-outliers) are Gaussian distributed and that the outliers have an arbitrary Laplace distribution. We can approximate the probability density function of the residuals with [29, 70]

$$f(r) = \begin{cases} \frac{1-\varepsilon}{\sqrt{2\pi}} e^{-\frac{r^2}{2}} & \text{if } |r| < a \\ \frac{1-\varepsilon}{\sqrt{2\pi}} e^{-(a|r|) - \frac{a^2}{2}} & \text{otherwise} \end{cases}.$$  \hfill (10.45)
The cost function is
\[
\varrho(r) = \begin{cases} 
    r^2 & \text{for } |r| < a \\
    a \cdot |r| - \frac{r^2}{2} & \text{otherwise}
\end{cases}
\] (10.46)

with the associated M estimator
\[
\Psi(r_j) = \max \left[ -a, \min(r_j, a) \right],
\] (10.47)

where \(a\) is the threshold for detecting outliers. In order to adapt the outlier detection to the image difference signal \(\Delta I\), we select a proportional to \(\sigma_{\Delta I}\) (10.37).

Often, the probability density function \(f(r)\) is unknown. Therefore, heuristic solutions for \(\Psi(r)\) such as the cost function \((1 - r^2/b^2)^2\) according to Turkey were found [38, 70]:
\[
\Psi(r_j) = \begin{cases} 
    r_j \cdot \left(1 - \frac{r_j^2}{b^2}\right)^2 & \text{if } |r_j| < b \\
    0 & \text{otherwise}
\end{cases}
\] (10.48)

The cost \((1 - r^2/b^2)^2\) increases to 1 when \(|r|\) decreases. Observation points with \(|r| \geq b\) are excluded from the current iteration; \(b\) is the threshold for detecting outliers. In order to adapt the outlier detection to the image difference signal \(\Delta I\), we select \(b\) proportional to \(\sigma_{\Delta I}\) (10.37).

The shape difference between the model object and the real object can be modeled by means of a spatial uncertainty of an observation point along the line of sight. This can be considered using a Kalman filter during motion estimation [39].

**Experimental Results**

Ideally, each iteration \(i\) of our motion estimation would consist of four steps. First, we solve (10.35) with all observation points. Then we select the observation points that fulfill the criteria for robust motion estimation. In the third step, we estimate the motion parameters using (10.35) again with these selected observation points. Finally, we use these parameters for motion compensation according to (10.17). In order to avoid solving (10.35) twice, we use the observation points that fulfilled the criteria for robust estimation in iteration \(i - 1\). Hence, each iteration \(i\) consists of three steps: (1) solve (10.35) using the observation points selected in iteration \(i - 1\), (2) motion compensate the model object with the estimated motion parameters, and (3) select the new observation points to be used in iteration \(i + 1\).

In order to evaluate the motion estimation algorithm and the robust estimation methods, we test the algorithm on synthetic image pairs that were generated with known parameters [51]. First, we create a model object using the test sequence Claire (Figure 10.10f). We create the first test image by projecting the model image into the image plane. Before synthesizing the second test image, we move the model object and change its facial expression, simulating motion and model failures, respectively. Finally, we add white Gaussian noise with variance \(\sigma_n^2\) to the test images. In the following tests, we use the model object that corresponds to the first test image. We estimate the motion parameters that resulted in the second test image.

As a first quality measure, we compute the average prediction error variance \(\sigma_{\text{diff}}^2\) inside the model object silhouette between the motion-compensated model object and the second test image. As a second quality measure, we compute the error of the model object position, measured as the average position error \(d_{\text{mot}}\) of all vertices between the estimated position \(\hat{P}^{(n)}\) and the correct position \(P^{(n)}\):
\[
d_{\text{mot}} = \frac{1}{N} \sum_{n=1}^{N} \left| P^{(n)} - \hat{P}^{(n)} \right|.
\] (10.49)
With (10.49), we can capture all motion estimation errors. The smaller $d_{\text{mot}}$ gets, the better the estimator works in estimating the true motion between two images. Estimation of true motion is required in order to be able to track an object in a scene. We do not compare directly the estimated motion parameters because they consist of translation and rotation parameters that are not independent of each other, thus making it difficult to compare two motion parameter sets.

Figure 10.20 shows the prediction error $\sigma_{\text{diff}}^2$ in relation to the image noise for three thresholds. Using the threshold $T_{ST} = \infty$ according to (10.36) allows us to consider every observation point. We use only observation points with a small luminance difference when setting $T_{ST} = 1$. Alternatively, we can use (10.38). Hence, we select only observation points that indicate a small local motion. As can be seen, removing the outliers from the measurement data reduces the prediction error. Due to the model failure, the prediction error variance $\sigma_{\text{diff}}^2$ does not decrease to 0. As can be seen in Figure 10.20, the prediction error increases with the image noise.

If we compare the average position error $d_{\text{mot}}$, we see it decreases significantly when we use the criteria according to (10.36) and (10.38) (Figure 10.21). $d_{\text{mot}}$ is an important criterion for estimating the true motion of an object. According to the experiments shown in Figure 10.21, using $\Delta I(q(j))$ with (10.36) and the threshold $T_{ST} = 1$ as the control criterion results in the smallest position error $d_{\text{mot}}$.

Using an M estimator further decreases the position error $d_{\text{mot}}$. Using the probability density function (10.45) with

$$a = \sigma_{\Delta I} \cdot T_{ST},$$

and the prediction error variance $\sigma_{\Delta I}^2$ of all observation points during iteration $i$, gives the best results for $T_{ST} = 0.2$. The most precise estimation was measured using the cost function according to Turkey, (10.48), with

$$b = \sigma_{\Delta I} \cdot T_{ST},$$

the prediction error variance $\sigma_{\Delta I}^2$ of all observation points during iteration $i$, and $T_{ST} = 1$. The position error $d_{\text{mot}}$ is not influenced much by image noise. This is because the noise is Gaussian and the model object covers a relatively large area of the image (30% for Claire).
FIGURE 10.21
Average deviation of control point position $d_{mot}$ according to (10.49) as a function of image noise $\sigma_n$. The curves were created using different criteria: $V_{\text{max}}$ according to (10.19), $T_{ST}$ according to (10.17), $a$ according to (10.45) and (10.50), $b$ according to (10.48) and (10.51), and $\sigma_{\Delta I}$ according to (10.37).

Segmentation into Components

The initial segmentation of moving objects into components was developed by Busch [6]. After motion compensating the entire rigid model object, the segmentation algorithm clusters neighboring triangles with similar 2D motion parameters. If these clusters allow for an improved motion compensation, a model object is subdivided into flexibly connected components. This decision is based on the evaluation of two consecutive images only.

In [39], Martínez proposes a different approach to segmenting an object into components. The 3D motion is measured for each triangle. In order to achieve a reliable estimate for the triangles, a Kalman filter modeling model object shape errors and the camera noise is adopted. Triangles with similar 3D motion are stored in a cluster memory. As soon as a cluster in the memory is stable for several images, this cluster is used to define a component of the object. This algorithm yields segmentation of persons into head, shoulders, and arms. Further improvements in motion estimation are achieved by enforcing spatial constraints due to spherical joints between components.

After segmenting an object into rigid components, 3D motion is estimated iteratively for each individual component as well as for the entire object.

10.4.3 MF Objects

MF objects are not related to real objects. They are just used to cover the deficiencies of the object and illumination models as well as estimation errors. Therefore, MF objects are always detected at the end of image analysis. This procedure can be seen as the final verification step of image analysis. According to the scene model, MF objects exist only in the model image plane.

Detection of MF Objects

MF objects are estimated by comparing the current real image with the model image $s_k^*$, which is synthesized using previously transmitted color parameters $S_k^*$ and the current motion and shape parameters $A_{k+1}^M$ and $M_{k+1}^M$, respectively (Figures 10.16 and 10.18).
this comparison, we will segment those image areas that cannot be described with sufficient subjective quality using MC objects as defined by the source model. Each MF object is described by its 2D silhouette and the color parameters inside the silhouette.

The detection of MF objects implies a receiver model. The following list gives some qualitative properties of the receiver model. It is assumed that the subjective image quality is not disturbed by:

1. Camera noise
2. Small position errors of the moving objects
3. Small shape errors of the moving objects
4. Small areas with erroneous color parameters inside a moving object

The errors listed as items (2) to (4) are referred to as geometrical distortions. Properties of the human visual system such as the modulation transfer function and spatiotemporal masking are not considered.

The following algorithm implicitly incorporates the above-mentioned assumptions. The difference image between the prediction image \( s^{*}_{k+1} \) and the current image \( s_{k+1} \) is evaluated by binarizing it using an adaptive threshold \( T_e \) such that the error variance of the areas that are not declared as synthesis errors is below a given allowed noise level \( N_e \). \( N_e = 6/255 \) is a commonly used threshold. The resulting mask is called the synthesis error mask. Figure 10.22a and b show a scaled difference image and the resulting synthesis error mask, respectively.

The synthesis error mask marks those pels of image \( s^{*}_{k+1} \) which differ significantly from the corresponding pels of \( s_{k+1} \). Since the areas of synthesis errors are frequently larger than 4% of the image area, it is not possible to transmit color parameters for these areas with a sufficiently high image quality (i.e., visible quantization errors would occur). However, from a subjective point of view it is not necessary to transmit color parameters for all areas of synthesis errors. Due to the object-based image description, the prediction image \( s^{*}_{k+1} \) is subjectively pleasant. There are no block artifacts, and object boundaries are synthesized properly.

There are two major reasons for synthesis errors. First of all, synthesis errors are due to position and shape differences between a moving real object and its corresponding model object. These errors are caused by motion and shape estimation errors. They displace contours in the image signal and will produce line structures in the synthesis error mask. Due to the feedback of the estimated and coded motion and shape parameters into image analysis (Figure 10.1), these estimation errors tend to be small and unbiased and they do not accumulate. Therefore, it is reasonable to assume that these errors do not disturb subjective image quality. They are classified as geometrical distortions. As a simple detector of geometrical distortions, a median filter of size 5 \( \times \) 5 pel is applied to the mask of synthesis errors (Figure 10.22c).

Second, events in the real world that cannot be modeled by the source model will contribute to synthesis errors. Using the source model R3D, it is not possible to model changing human facial expressions or specular highlights. Facial expressions in particular are subjectively important. In order to be of subjective importance, it is assumed that an erroneous image region has to be larger than 0.5% of the image area (Figure 10.22c). Model failures are those image areas where the model image \( s^{*}_{k+1} \) is subjectively wrong (Figure 10.22d). Each area of model failure is modeled by an MF_R3D object, defined by color and 2D shape parameters (Figure 10.23a and b). Applying the F3D object model to the MF_R3D objects (Figure 10.18) can compensate for some of the synthesis errors such that the MF_F3D objects tend to be smaller (Figure 10.23).
10.5 Optimization of Parameter Coding for R3D and F3D

The task of parameter coding is the efficient coding of the parameter sets motion, shape, and color provided by image analysis. Parameter coding uses a coder mode control to select the appropriate parameter sets to be transmitted for each object class. The priority of the parameter sets is arranged by a priority control.

10.5.1 Motion Parameter Coding

The unit of the estimated object translation $T = (t_x, t_y, t_z)^T$ is pel. The unit of the estimated object rotation $R_C = (R_x^{(C)}, R_y^{(C)}, R_z^{(C)})^T$ is degree. These motion parameters are PCM coded by quantizing each component with 8 bit within an interval of $\pm 10$ pel and degrees, respectively. This ensures a subjectively lossless coding of motion parameters.

FIGURE 10.22
Detection of model failures: (a) scaled difference image between real image $s_{k+1}$ and model image after motion and shape compensation $s_{k+1}^*$; (b) synthesis error mask; (c) geometric distortions and perceptually irrelevant regions; (d) mask $\text{MF}_{\text{R3D}}$ with model failures of the source model rigid 3D object.
FIGURE 10.23
Detection of model failures: MF_R3D objects with (a) shape and (b) color parameters. After
the source model F3D is applied to the MF_R3D objects, the MF_F3D objects are detected
with the (c) shape and (d) texture parameters. (a) is an enlargement of Figure 10.22d.

10.5.2 2D Shape Parameter Coding

Since the model object shape is computed and updated from its silhouette, shape parameters
are essentially 2D. The principles for coding the shape parameters of MF and MC objects
are identical. Shape parameters are coded using a polygon/spline approximation developed
by Hötter [24]. A measure $d_{\text{max}}$ describes the maximum distance between the original and
approximated shape. First, an initial polygon approximation of the shape is generated using
four points (Figure 10.24a). Where the quality measure $d^*_{\text{max}}$ is not satisfied, the approximation

is iteratively refined through insertion of additional polygon points until the measure fulfills
$d_{\text{max}} \leq d^*_{\text{max}}$ (Figure 10.24b). In case the source model F2D is used, we check for each
line of the polygon, whether an approximation of the corresponding contour piece by a spline
approximation also satisfies $d_{\text{max}}^*$. If so, the spline approximation is used, giving a natural shape approximation for curved shapes (Figure 10.25). In order to avoid visible distortions at object boundaries, MC objects are coded with $d_{\text{max}}^* = 1.4$ pel. Experimental results showed that MF objects should be coded with $d_{\text{max}}^*$ about 2.1 pel in order to minimize the overall bit rate required for coding MF objects [46].

The coordinates of the polygon points are coded relative to their perspective predecessor. In the case of the source model F2D, the curve type line/spline is coded for each line of the polygon.

The data rate for coding shape parameters of MC objects is cut to half by using the motion-compensated coded silhouette of the last image as a prediction of the current silhouette. Starting with this approximation, only shape update parameters have to be transmitted.

### 10.5.3 Coding of Component Separation

The split of an object into components is defined on a triangle basis. Whenever a new component is defined by the encoder, its shape is encoded losslessly with a flag for each visible triangle. The encoder and decoder then define the shape of the component by connecting the visible triangles with the ones that they occlude at the back of the object.

### 10.5.4 Flexible Shape Parameter Coding

A list of all currently visible control points having flexible shape parameters $S_{\ell}^n \neq 0$ is transmitted using a run-length code. In a second step, the components of the corresponding vectors $S_{\ell}^n$ are linearly quantized using 16 representation levels within an interval of ±5 pel. The quantized vector components are entropy coded.

### 10.5.5 Color Parameters

Conventional DCT is not suitable for the coding of color parameters of arbitrarily shaped regions. New algorithms have been developed for this application [17, 61]. Here the special type of DCT for arbitrarily shaped regions developed by Gilge [17] is improved by applying a segmentation of the color parameters into homogeneous regions prior to transform coding [47]. The segmentation is based on the minimum spanning tree [42] using the signal variance as criterion. The boundaries of the regions are coded using a chain code [16]. The DCT coefficients are quantized with a linear quantizer of signal-dependent step size. The advantage of this scheme using segmentation prior to transform coding is that errors due to coarse quantization
are mainly concentrated at the boundaries of the segmented regions, where they are less visible
due to masking of the human visual system in areas of high local activity.

10.5.6 Control of Parameter Coding

Due to limited data rate, a transmission of all parameter sets cannot be guaranteed. Coder
control is used to overcome this difficulty. It consists of coder mode control and priority control.
Coder mode control selects the relevant parameter sets and coder adjustments for each object,
and priority control arranges these parameter sets for transmission (see Table 10.1).

Depending on the model object class MF or MC, the coder mode control selects two param-
eter sets for transmission. For MC objects, only motion $A_{k+1}(m)$ and shape update parameters
$M_{k+1}(m)$ are coded. Coding of color parameters is not necessary because the existing color pa-
rameters $S_k(m)$ of the model objects are sufficient to synthesize the image properly. 2D shape
parameters, defining the location of the model failures in the image plane, and color parameters
are coded for MF objects.

Priority control guarantees that the motion parameters of all MC objects are transmitted
first. In a second step, the shape parameters of the MC objects are transmitted. Finally, the
shape and color parameters of the MF objects are transmitted until the available data rate is
exhausted.

10.6 Experimental Results

The object-based analysis–synthesis coder based on the source models R3D and F3D is
applied to the test sequences Claire [10] and Miss America [5], with a spatial resolution
corresponding to CIF and a frame rate of 10 Hz. The results are compared to those of an
H.261 coder [8, 9] and OBASC based on the source model F2D as presented by Hötter [26]–
[28]. As far as detection of model failures and coding of shape parameters are concerned, the
same algorithms and coder adjustments are applied. Parameter coding aims at a data rate of
approximately 64 kbit/s. However, the bit rate of the coder is not controlled and no buffer is
implemented. In the experiments, the allowed noise level $N_e$ for detection of model failures is
set to $6/255$. Color parameters of model failures are coded according to Section 10.5.5 with a
peak signal-to-noise ratio (PSNR) of 36 dB. In all experiments the coders are initialized with
the first original image of the sequence (i.e., the frame memory is initialized with the first
original image for the block-based coder H.261). For the two object-based analysis–synthesis
coders, the model object Background in the memory for parameters is initialized with the first
original image.

For head and shoulder scenes the 3D model object is usually divided into two to three
components. Applying the estimated motion parameter sets to the model object gives a natural
impression of object motion. This indicates that the estimated motion parameters are close to
the real motion parameters and that the distance transform applied to the object silhouette for
generating the 3D model object shape is suitable for the analysis of head and shoulder scenes.

The area of rigid model failures $MF_{R3D}$ is on average less than 4% of the image area
(Figure 10.26). Generalizing this, for head and shoulder scenes the rigid model failures can
perhaps be expected to be less than 15% of the moving area. The exact figures of model failure
area as related to moving area are 12% for Claire and 7% for Miss America. The test sequence
Claire seems to be more demanding, due to the fast rotation of the subject’s head, whereas
Miss America’s motion is almost 2D.
FIGURE 10.26
Area of rigid model failures $M_{R3D}$ in pel for the test sequence Claire. The total area is 101,376 pel. The average area of model failures is 3.5% of the image area.

Table 10.2 compares the average bit rate for the different parameter sets motion, shape, and color and the source models F2D, R3D, and F3D. Coding of the head and shoulder test sequences Claire and Miss America and the MPEG-4 test sequence Akiyo will not exceed the data rates given in Table 10.2. The source models F2D and R3D need approximately the same data rate. Due to the displacement vector field, OBASC based on the source model F2D requires a relatively high amount of motion information. Shape parameters include the shape of MC and MF objects. Shape parameters of MC objects require similar data rates for both source models. However, the source model F2D causes only a few large MF objects, whereas the source model R3D causes smaller but more MF objects. This larger number of $M_{R3D}$ objects is due to the applied source model assuming rigid shapes. Shape differences between real and model objects as well as small flexible motion on the surface of real objects cannot be compensated for. These effects cause small local position errors of the model objects. If texture with high local activity is displaced for more than 0.5 pel, model failures are detected due to the simple filter for the detection of geometric distortions. Since these small position errors can be compensated for when using the source model F2D, the overall data rate for shape parameters is 750 bit higher for the source model R3D. Since this is due to the more local motion model of F2D, we have to also look at comparing the sum of motion and shape data rates. Here, the source model R3D requires 7.5% fewer bits than the source model F2D.

Table 10.2  Average Bit Rate of Parameter Sets for Different Source Models

<table>
<thead>
<tr>
<th>Source Model</th>
<th>Motion: $R_A$ (bit/frame)</th>
<th>Shape of MC Objects: $R_{M,MC}$ (bit/frame)</th>
<th>Shape of MF Objects: $R_{M,MF}$ (bit/frame)</th>
<th>Area of MF Objects and Uncovered Background (% of image area)</th>
</tr>
</thead>
<tbody>
<tr>
<td>F2D</td>
<td>1100</td>
<td>900</td>
<td>900</td>
<td>4%</td>
</tr>
<tr>
<td>R3D</td>
<td>200</td>
<td>500</td>
<td>1150</td>
<td>4%</td>
</tr>
<tr>
<td>F3D</td>
<td>200</td>
<td>950</td>
<td>1000</td>
<td>3%</td>
</tr>
</tbody>
</table>

*Note: The coders use the same algorithm for detection of model failures.*

When comparing F3D with R3D, we notice that the average area of MF objects decreases from 4 to 3%. For CIF sequences, this results in bit savings of at least 1000 bit/frame for the texture parameters of MF objects. At the same time, the data rate for MF object shape decreases from 1150 to 1000 bit/frame. This is mainly due to the smaller number of MF objects in the case of F3D. Since the use of F3D requires the transmission of the flexible shift vectors as an additional dataset, the rate for MC object shape parameters increases by 450 bit/frame.
This indicates that by spending an additional 450 bit/frame on the shape parameters of an MC object, we save 150 bit/frame on the MF object shape and reduce the area of MF objects by 1%. Therefore, the bit savings is significantly higher than the costs for this additional parameter set of F3D.

Figure 10.27 shows part of the 33rd decoded frame of the test sequence Claire using the source models F2D, R3D, F3D, and H.261. Subjectively, there is no difference between the source models F2D, R3D, and F3D. However, the source model F3D requires only 56 kbit/s instead of 64 kbit/s. When compared to decoded images of an H.261 coder [9], picture quality is improved twofold (Figure 10.28). At the boundaries of moving objects, no block or mosquito artifacts are visible, due to the introduction of shape parameters. Image quality in the face is improved, because coding of color parameters is limited to model failures, which are mainly located in the face. Since the average area of model failures (i.e., the area where color parameters have to be coded) covers 4% of the image area, color parameters can be coded at a data rate higher than 1.0 bit/pel. This compares to 0.1 to 0.4 bit/pel available for coding of color parameters with an H.261/RM8 encoder.

FIGURE 10.27
Part of the 33rd decoded frame of test sequence Claire at a data rate of 64 kbit/s: (a) block-based hybrid coder H.261 (RM8), (b) F2D, (c) R3D, (d) F3D at 56 kbit/s.

10.7 Conclusions

In this chapter the concept and implementation of an object-based analysis–synthesis coder based on the source models of moving rigid 3D objects (R3D) and moving flexible 3D objects (F3D) aiming at a data rate of 64 kbit/s have been presented. An OBASC consists of five parts: image analysis, parameter coding and decoding, image synthesis, and memory for parameters. Each object is defined by its uniform 3D motion and is described by motion, shape, and color parameters. Moving objects are modeled by 3D model objects.
The goal of image analysis is to arrive at a compact parametric description of the current image of a sequence, taking already transmitted parameter sets into account. Image analysis includes shape and motion estimation as well as detection of model failures. Moving objects are segmented using temporal change detection and motion parameters. The algorithm for estimating these motion parameters is based on previous work on gradient-based motion estimation. A new set of equations relating the difference signal between two images to 3D motion parameters has been established. Using robust motion estimation algorithms enables us to track rigid as well as flexible objects such as head and shoulders throughout a video scene, thus enabling an efficient object-based video coder. The 3D shape of a model object is computed by applying a distance transformation, giving object depth, to the object silhouette. Since the estimated motion parameters applied to these 3D model objects give a natural impression of motion, this distance transform is very suitable for analysis of head and shoulder scenes.

Those areas of an image that cannot be modeled by the applied source model are referred to as model failures and are modeled by MF objects. They are described by color and 2D shape parameters only. Model failures are detected taking subjective criteria into account. It is assumed that geometrical distortions such as small position and shape errors of the moving objects do not disturb subjective image quality. Due to these subjective criteria, the average area of model failures is less than 4% of the image area for typical videophone test sequences.

Flexible shift parameters of the source model F3D are estimated only for those parts of an object that cannot be described using the source model R3D. This limits the additional data
rate required for the flexible shift parameters and increases the overall efficiency of this source model F3D over R3D.

With respect to coding, shape and color parameters are coded for MF objects, whereas motion and shape update parameters have to be coded for MC objects. Motion parameters are PCM coded and shape parameters are coded using a polygon approximation. Prior to coding, color parameters are segmented into homogeneous regions. Then a DCT for arbitrarily shaped regions is applied.

The presented coder has been compared to OBASC based on the source model of moving flexible 2D objects (F2D). With regard to typical head and shoulder videophone test sequences, it is shown that the picture quality at the average bit rate of 64 kbit/s is the same regardless of whether the source model R3D or F2D is applied. When using F3D, the data rate shrinks from 64 to 56 kbit/s for the same picture quality. When compared to images coded according to H.261, there are no mosquito or block artifacts because the average area for which color parameters are transmitted is 10% of the image area for H.261 and 4% for OBASC. Therefore, OBASC allows coding of color parameters for MF objects with a data rate higher than 1.0 bit/pel. At the same time, MC objects are displayed without subjectively annoying artifacts.

This chapter has demonstrated the feasibility of OBASC using 3D models. As shown in [31], this coder can be used as a basis for knowledge-based and semantic coders. However, image analysis as presented here is not yet able to describe with sufficient accuracy scenes with a lot of motion, such as gesticulating hands. In [39], the 3D motion estimation algorithm is extended to enable the segmentation of flexibly connected rigid objects and components. In [40], an OBASC with an image analysis is presented that enables camera motion compensation. The concept of MF detection is viable for any coder that uses smooth motion vector fields and knows the approximate location of object boundaries [54, 66]. It enables reduction of the data rate of a block-based coder without decreasing the subjective image quality.

References


