ELSEVIER

Signal Processing: *Image Communication* 15 (2000) 729–766

SIGNAL PROCESSING: IMAGE COMMUNICATION

www.elsevier.nl/locate/image

A quadratic motion-based object-oriented video codec☆

Y. Yemez, B. Sankur*, E. Anarım

Department of Electrical and Electronics Engineering, Boğaziçi University, 80815, Bebek, İstanbul, Turkey

Received 15 September 1997

Abstract

A novel motion-based object-oriented codec for video transmission at very low bit-rates is proposed. The object motion is modeled by quadratic transform with coefficients estimated via a nonlinear quasi-Newton method. The segmentation problem is put forward as a constrained optimization problem which interacts with the motion estimation process in the course of region growing. A context-based shape coding method which takes into account the image synthesis error as well as the geometric distortion, is also proposed. Quantitative and subjective performance results of the codec on various test sequences illustrate the superior performance of the method.

Keywords: Object-oriented video coding; Motion segmentation; Region growing; Quadratic transform; Shape polygonization

1. Introduction

The research activities for very low bit-rate video transmission, in the context of model-based coding, have been centered around the development of the new international standard, so-called MPEG-4 [24]. This standard, besides high compression, also addresses content-based functionalities such as interactivity (e.g., multimedia data access, manipulation) and scalability (e.g., user or automated selection of decoded quality of objects in the scene, database browsing at different qualities). Therefore novel video coding schemes are desired, which can address the content of the video by differentiating the objects or the regions of interest in the scene. These schemes also exploit in general the properties of the human visual system, as well as the inherent temporal redundancy using more sophisticated motion estimation.

Conventional schemes such as block-based H.261/263 standard, are mostly independent of the scene content. As far as model-based algorithms are concerned, on one side stand the semantic-based schemes which are committed to specific content, such as those based on head-and-shoulder wireframe models. On the other side, the object-oriented schemes make no a priori assumptions about the scene content, and are

☆This work was supported by TÜBİTAK under contracts EEEAG-139 and EEEAG-83.

* Corresponding author. Fax: 90-212-2872465.

E-mail address: sankur@boun.edu.tr (B. Sankur).

0923-5965/00/$ - see front matter
PII: S0923-5965(99)00017-X

Nomenclature

A	motion parameter set	$R(v_k)$	region lying between the shape and the polygon edge $P(v_{k-1})$
C	color parameter set		
D	geometric shape distortion	$\mathscr{R}$	image plane
$D_{\max}$	geometric shape distortion tolerance towards inside	s_l	boundary pixel
		s_{v_k}	boundary pixel associated to v_k
$D'_{\max}$	geometric shape distortion tolerance towards outside	S	shape parameter set
		t_l^i	band pixel associated to $T(s_l)$
E	mean-square synthesis error	T	band
G	overall coding cost	$T(s_l)$	subset of T associated to s_l
J	overall modeling error	T_{CH}	error threshold of change detection
$\boldsymbol{L}$	labeling map	T_{MC}	error threshold of segmentation
$\boldsymbol{p}$	motion parameter vector	v_k	vertex pixel
$\mathscr{P}$	image plane partition	V	ordered set of vertices
$P(v_{k-1}, v_k)$	polygon edge	γ	shape bit-rate for a polygon edge
R	object region	Γ	shape bit-rate for a polygon

therefore applicable to a more general class of images. Object-oriented methods model the real world in terms of objects which are specified by three sets of parameters defining the motion, shape and color (texture) information. In this respect object-oriented codecs have gained prominence since they both allow for enhanced functionality and also they have the potential for improved rendition using advanced motion models.

The three important tasks in an object-oriented coding scheme are object modeling, image analysis and synthesis, and parameter coding, as shown in the block diagram of Fig. 1. Image analysis purports to segment the scene into objects. The object modeling part refers to the definition of objects and determines the goal-directed criteria of the segmentation process. Segmentation generally yields arbitrarily shaped and textured regions corresponding to differently moving objects. Finally, parameter coding aims to coding of the shape, color and motion parameters of resulting objects, whose bit budget should be efficiently shared among these three types of parameters.

The most crucial part of an object-oriented coder is the analysis part, in which the scene is segmented into objects under the chosen object model. The main approaches in the literature can be grouped under two categories:

Temporal segmentation. The goal of temporal segmentation in an object-oriented coder is to subdivide the scene into regions each of which corresponds to an object with a different motion characteristic. In [21], the segmentation depends purely on motion information, and proceeds hierarchically from larger objects to smaller ones. The scene is first divided into background and changed regions. The changed region is further processed and segmented into model compliance, model failure and uncovered regions [14]. Model compliance regions correspond to objects for which the motion can be modeled successfully by the chosen object model, and thus can be compensated without further color coding. On the other hand, model failure and uncovered regions fail to satisfy the object model, and therefore are subject to color coding. The various object models utilized in such a scheme are 2D rigid objects with 3D motion [21], 2D flexible objects with 2D motion [12], 3D rigid objects with 3D motion [21], 3D flexible objects with 3D motion [23]. Although this scheme, in terms of bit-rate-quality tradeoff, works well for head-shoulder scenes of videophone sequences, for which it has been designed for, it may fail for scenes of a different content.

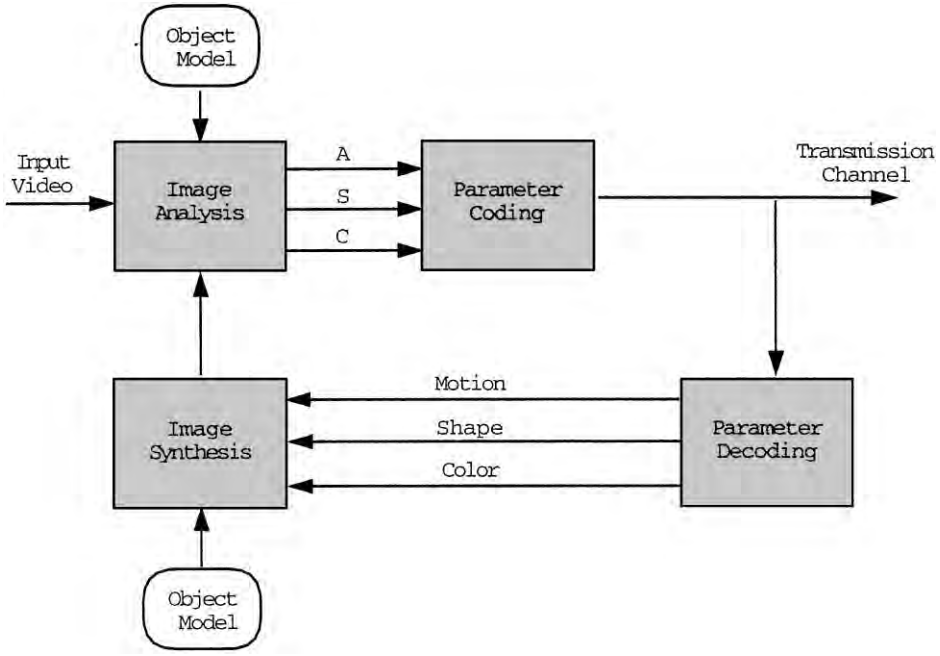

Fig. 1. Block diagram of an object-oriented coder.

Spatial segmentation. The goal of spatial segmentation in an object-oriented coder is to subdivide the scene into spatially homogeneous regions each of which corresponds to a different object. Thus, rather than the object motion, the spatial properties of the objects are taken into account for the segmentation. This leads to an image representation in terms of the primitives, contours and textures. The main motivation for this scheme is the observation that the most important part of the visual information is represented by contours [17,33], especially at high compression ratios.

In [26,27], morphological methods are employed in order to achieve a desirable segmentation. However, motion estimation and segmentation are treated completely independently, that is first the segmentation task is accomplished, followed by motion compensation of the texture and contour information. The advantage of this scheme, when compared to that in [14], is its generality in terms of the image content. In fact, the spatio-temporal segmentation makes use of both spatial and temporal characteristics of the objects, resulting in an oversegmentation which is recuperated by merging the regions with similar motion [19,10]. However, these schemes, which rely mainly on spatial information, need very accurate segmentation, and may potentially conflict with the real motion of the objects.

Our paper is aligned with coders in the "temporal segmentation" group as it is based on region growing motion segmentation approach which aims to distinguish and to code various objects in the video scene. This object-oriented codec [35,36] is mainly based on the framework of model compliance and model failure objects as presented in [12,14]. However, in contrast to [12], where a top-down hierarchy of motion segmentation was employed, we are using a bottom-up hierarchy. In other words, the object hierarchy is reverted in the sense that segmentation proceeds from smaller objects to larger ones. The segmentation in [12] proceeds from larger objects to smaller ones, and consists of only two levels of hierarchy so that it is applicable only to face-shoulder scenes of the videophone applications. Our scheme in this sense encompasses a more general class of video scenes. A second distinction of our work is that a more sophisticated

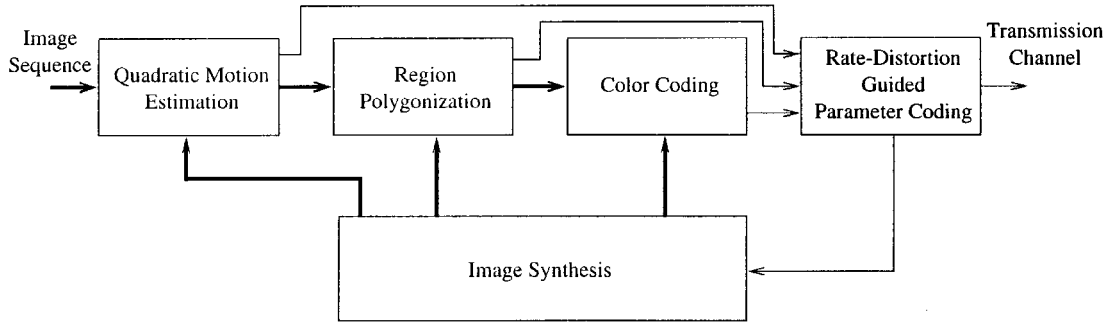

Fig. 2. Block diagram of the proposed motion-segmentation based coder (bold line: image flow; thin lines: parameter flow). The quadratic motion estimation block is further expanded in Fig. 6.

motion model is employed in the temporal segmentation. In fact, the motions of objects are modeled via quadratic transform, which enables a more general and flexible modeling as compared to the other lower order polynomial models used in the literature such as the affine transform and the pure translation. The quadratic model can represent 3-D motion of parabolic surfaces. A third contribution of the paper is the formulation of the segmentation problem in terms of the rate-distortion theory as an optimal description problem. The parametric representation resulting from the quadratic transform model avails us with the possibility of employing a region growing approach for the iterative solution of the segmentation problem. Finally a novel shape coding algorithm is developed for efficient polygonization of segment boundaries simultaneously taking into account both geometrical as well as image synthesis, and therefore motion representation distortions. The block diagram of the proposed motion-segmentation based coder is shown in Fig. 2.

The paper is organized as follows. In Section 2, the problem of quadratic object motion modeling is considered. In particular, the estimation procedure for the coefficients of the quadratic transform as a nonlinear optimization problem is addressed. The quasi-Newton optimization method, which is used for estimation, is presented in conjunction with a multiresolution and multiframe scheme. Section 3 addresses the analysis part of the algorithm, which aims to describe the scene in terms of moving objects with arbitrarily shaped and textured regions. The segmentation problem is formulated by a constrained optimization problem. The problem of finding the optimum description under the chosen object model is then solved suboptimally by an iterative region growing procedure. Section 4 addresses the coding issues of the shape, motion and color (texture) of the motion segments. A shape coding technique, which is based on polygon approximation and takes into account not only the geometric shape distortion, but also the synthesis error in terms of image intensity caused by the shape distortion, is proposed. In Section 5, the simulation results of the proposed approach are given and its performance on various test sequences is illustrated, both subjectively and quantitatively. Finally, Section 6 gives the concluding remarks.

2. Quadratic motion estimation

2.1. Object motion modeling

In most video coding schemes, the motion is modeled by a linear combination of 2D polynomial basis functions [5]. Translational and affine models, which correspond to the zeroth- and first-order polynomial transforms, are the most commonly used models in conventional and object-oriented coders. The use of

higher degree polynomial tranforms such as the quadratic transform [4,16], however, may lead to superior coding efficiencies since they are capable of modeling a broader class of object motions and of course, inherently contain the lower order representations.

The quadratic transform used for object motion modeling, relates the coordinates of an object pixel between two consecutive image frames at time instances $t-1$ and t. (This generic time difference denotes in this work the interval between two frames k apart, $k=1,2,\ldots$.) Thus, the image intensity $I(x,y,t)$ for an object region is mapped from $I(u(x,y),v(x,y),t-1)$, where the new coordinates are given by

$$\begin{aligned} u(x,y) &= a_1x^2 + a_2y^2 + a_3xy + a_4x + a_5y + a_6, \\ v(x,y) &= b_1x^2 + b_2y^2 + b_3xy + b_4x + b_5y + b_6. \end{aligned} \tag{1}$$

The use of a quadratic transform for motion modeling is meaningful only under certain physical conditions of the 3D geometric surface, of the 3D real motion of the objects, and of the projection of objects onto the 2D image plane. The quadratic transform gives an exact description of the 3D rotation, translation and scaling of an object with a parabolic surface under parallel projection [4]. Thus, the true object motion is modeled by

$$\boldsymbol{U} = \boldsymbol{R}\boldsymbol{X} + \boldsymbol{d}, \tag{2}$$

where $\boldsymbol{U} = (U,V,W)^{\mathrm{T}}$ and $\boldsymbol{X} = (X,Y,Z)^{\mathrm{T}}$ stand for object space coordinates, $\boldsymbol{R}$ is the rotation matrix, and $\boldsymbol{d}$ is the translation vector. Such a model inherently allows any linear deformation, but fails to model nonlinear deformations. On the other hand, the parabolic facet for surface modeling can be written as

$$Z = c_1X^2 + c_2XY + c_3Y^2 + c_4X + c_5Y + c_6. \tag{3}$$

The parabolic facet assumption (3) and the true object motion model (2) yield the quadratic transformation in terms of image plane coordinates under parallel projection. The quadratic transform is also a good approximation for the rigid motion and the linear deformation of an object with a quadratic surface under central projection [4] (see Fig. 3).

The quadratic transform models the real 3D world in terms of quadratic (parabolic) surfaces moving rigidly, or deforming linearly. Theoretically, the higher the degree of the polynomial representation, the more general is the model. One may then inquire whether even higher degree models would not be advantageous. In general, higher degree models will not necessarily result in a more efficient coding. First, higher polynomial representations need more parameters, hence increase the bit-rate. Second, with higher order polynomials the dimensionality and the nonlinearity of the optimization process employed for the parameter

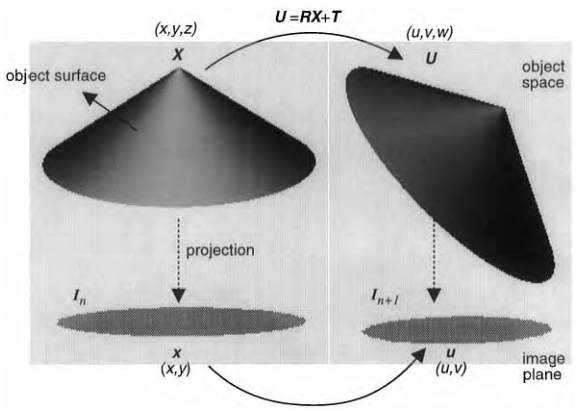

Fig. 3. Physical interpretation of the quadratic transform.

estimation increase. Our experiments have in fact shown that using third-order representations, for example, may yield even less accurate results compared to those of the quadratic transform. Thus, we believe that the quadratic transform in (1) represents the best compromise between modeling sophistication and coding gain.

2.2. Motion estimation

The motion estimation, in the case of continuous parameter space, is carried out via an optimization process that utilizes spatio-temporal properties of the image intensity, and is based on differential methods. The error function in motion estimation problem is given by the mean-squared error

$$E(\boldsymbol{p}, R) = \sum_{\boldsymbol{x} \in R} \mathrm{DFD}^2(\boldsymbol{x}, \boldsymbol{p}), \tag{4}$$

where R is the support region of the object. The parameter vector $\boldsymbol{p}$ is determined by the quadratic transform coefficients in (1), $\boldsymbol{p} = (a_1, \ldots, a_6, b_1, \ldots, b_6)^{\mathrm{T}}$, DFD denotes the displaced frame difference given by $\mathrm{DFD}(\boldsymbol{x},\boldsymbol{p}) = I(\boldsymbol{x}, t) - I(\boldsymbol{u}, t-1)$, where $\boldsymbol{u}$ is given by the quadratic transform of $\boldsymbol{x}$. In the sequel, the motion parameter set will be indicated by the symbol $A = a_k, b_k$, $k = 1, \ldots, 6$. For example, A_i will be used to denote the set of 12 motion parameters of the region R_i. In other words, the motion parameters will be denoted by $\boldsymbol{p}$ and A, respectively in the context of vector operations and set operations.

The error function in (4) is a nonlinear function, and the minimization of this function is possible only via nonlinear iterative differential methods such as the quasi-Newton method [25,7], as used in our implementation.

Differential algorithms start with an initial value, which is then iterated towards the optimum by accumulating the information. Newton's method is based on a quadratic model. To illustrate the case let us consider any arbitrary N-dimensional nonlinear function $f(\cdot)$ of $\boldsymbol{p}$. Using the truncated Taylor series expansion of $f(\boldsymbol{p})$ about the point $\boldsymbol{p}_k$, the quadratic model function $q_k(\boldsymbol{\delta})$ can be written, at iteration k, as

$$f(\boldsymbol{p}_k + \boldsymbol{\delta}) \approx q_k(\boldsymbol{\delta}) = f(\boldsymbol{p}_k) + \boldsymbol{g}_k^{\mathrm{T}}\boldsymbol{\delta} + \tfrac{1}{2}\boldsymbol{\delta}^{\mathrm{T}}\boldsymbol{H}_k\boldsymbol{\delta}, \tag{5}$$

where $\boldsymbol{\delta} = \boldsymbol{p} - \boldsymbol{p}_k$. In (5), $\boldsymbol{g}_k$ denotes the gradient of the function, and $\boldsymbol{H}_k$ is the Hessian matrix.

One disadvantage of Newton's method is the need for the second-order derivatives of the function, and the other is that the Hessian matrix $\boldsymbol{H}$ must be positive definite to assure convergence. The latter disadvantage can be avoided by *Newton's method with line search* for finding successive directions along which line minimizations are performed so that the current point $\boldsymbol{p}_k$ converges towards the minimum. The so-called "quasi-Newton" method, which is also based on line minimizations, avoids the computation of the Hessian matrix at each iteration by updating the inverse Hessian matrix $\boldsymbol{H}_k^{-1}$ in terms of the value of the function and its first-order derivatives. The kth iteration of the quasi-Newton algorithm can be summarized by the following three steps:

1. set $\boldsymbol{s}_k = -\boldsymbol{H}_k^{-1}\boldsymbol{g}_k$,
2. line minimization along $\boldsymbol{s}_k$ so that $\boldsymbol{p}_{k+1} = \boldsymbol{p}_k + \alpha_k\boldsymbol{s}_k$,
3. update $\boldsymbol{H}_k^{-1}$ into $\boldsymbol{H}_{k+1}^{-1}$,

where $\boldsymbol{s}_k$ corresponds to the direction along which the line minimization is carried out at iteration k, and α_k is a constant determined by line minimization. The BFGS (Broyden, Fletcher, Goldfarb, Shanno) method for the update of the Hessian matrix in [7], has been used in steps of the algorithm.

The first-order derivative of the error function $E(\boldsymbol{p}, R)$ in (4) can be computed numerically by

$$\frac{\partial E(\boldsymbol{p}, R)}{\partial \boldsymbol{p}} = -\sum_{\boldsymbol{x}_n \in R} 2(I(\boldsymbol{x}_n, t) - I(\boldsymbol{u}_n, t-1)) \frac{\partial I(\boldsymbol{u}_n, t-1)}{\partial \boldsymbol{u}} \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{p}}, \tag{6}$$

where $\boldsymbol{u}_n = (u_n, v_n)$ is given by (1) and

$$\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{p}} = \begin{bmatrix} x^2 & y^2 & xy & x & y & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & x^2 & y^2 & xy & x & y & 1 \end{bmatrix}, \tag{7}$$

$$\frac{\partial I(\boldsymbol{x}, t)}{\partial \boldsymbol{x}} = \begin{bmatrix} \dfrac{\partial I}{\partial x} & \dfrac{\partial I}{\partial y} \end{bmatrix}. \tag{8}$$

The partial derivatives with respect to x and y can be approximated by

$$\frac{\partial I(x, y, t)}{\partial x} \approx \frac{1}{2}(I(x+1, y, t) - I(x-1, y, t)), \tag{9}$$

$$\frac{\partial I(x, y, t)}{\partial y} \approx \frac{1}{2}(I(x, y+1, t) - I(x, y-1, t)). \tag{10}$$

A problem in computing the value of $E(R, \boldsymbol{p})$ and its derivatives is that the mapped coordinates u, v correspond to real-valued transforms in continuous space whereas $I(u, v)$ is defined only on a discrete grid. The image can be estimated on the grid coordinates by the bilinear interpolation [22]

$$I(u, v) = (1-\alpha)((1-\beta)I([u], [v]) + \beta I([u]+1, [v])) + \alpha((1-\beta)I([u], [v]+1) + \beta I([u]+1, [v]+1)), \tag{11}$$

where $([u], [v])$ are the integral parts and (α, β) are the fractional parts of the coordinate (u, v). The gradient vector in (6) must similarly be estimated on the coordinates (u, v) via interpolation on the original grid. Hence, Eq. (6) can alternatively be calculated by

$$\frac{\partial E(\boldsymbol{p}, R)}{\partial \boldsymbol{p}} = \sum_{\boldsymbol{x}_n \in R} 2(I(\boldsymbol{x}_n, t) - I(\boldsymbol{u}_n, t-1)) \frac{\partial I(\boldsymbol{x}_n, t)}{\partial \boldsymbol{x}} \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{u}} \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{p}}, \tag{12}$$

where

$$\left(\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{u}}\right)^{-1} = \begin{bmatrix} 2a_1 x + a_3 y + a_4 & 2a_2 y + a_3 x + a_5 \\ 2b_1 x + b_3 y + b_4 & 2b_2 y + b_3 x + b_5 \end{bmatrix}. \tag{13}$$

2.3. Multiscale estimation

The quasi-Newton methods are assured to converge globally in N iterations if the function $f(\boldsymbol{p})$ to be minimized is quadratic, regardless of the starting point $\boldsymbol{p}_0$. This is generally not the case for the error function in (4) which is set up for arbitrarily textured images. In other words, the error expression in (4) would be a quadratic function only when the related region corresponds to a quadratic surface. However, real-world images, especially textured areas do not form exactly quadratic surfaces. Thus, there may exist many spurious minima on the trajectory between the starting point to the global minimum. In order to prevent the quasi-Newton algorithm from getting stuck in a local minimum, we recourse to the following measures: (a) proper initialization of $\boldsymbol{p}$, (b) multiresolution scheme, and (c) tracking of motion in skipped frames. Let us

emphasize once more that with the multiresolutional scheme the lower bands are less textured, and their regions can be more easily modeled as quadratic surfaces. This is the reason why better initial estimates can be obtained. In summary, the interplay of spatial and temporal multiresolution guarantees and speeds up the convergence of the quasi-Newtonian methods.

The proper initialization of $\boldsymbol{p}$, recalling (1), is based on the following observation:

$$u(0,0) = a_6, \qquad v(0,0) = b_6. \tag{14}$$

Thus the coefficients of the zeroth-order terms corresponding to the translational part are determined by the displacement of the point $(0,0)$, i.e., the geometric center of the support region. The vertical and horizontal displacements of the centroid of a region R, which can be estimated by a classical block matching technique, correspond to the initial values of a_6 and b_6, respectively.

A second measure to improve convergence is a multiresolution scheme as in [15,5,16], which spatially smooths the consecutive images and reduces the number of the minima arising from the high-frequency components. The estimates obtained from the smoothed images can then be used for a more accurate estimate in the higher bandwidth images.

Thirdly, the motion information in the skipped frames, which is usually neglected in coding schemes, can also be incorporated so that the motion parameters evolve temporally through the skipped frames and converge towards the global minimum. This is needed especially when there exists severe motion as a consequence of frame skipping.

Consider the problem of motion estimation between two P-frames (predictive coded frames), I_n and $I_{n-\tau}$, so that there are $\tau - 1$ S-frames (skipped frames) in between (Fig. 4). Notice that in this algorithm the motion estimation proceeds in the backward sense. The skipped frames are only used by the encoder in order to improve the performance of the estimation, but they are not coded and transmitted to the receiver. The motion of each object in the scene between the P-frames is determined by a quadratic transform coefficient vector which can be denoted by $\boldsymbol{p}(\tau)$. Suppose that the motion of an object between the P-frame I_n and a skipped frame I_{n-k}, $k < \tau$ has been estimated. This already estimated parameter vector $\boldsymbol{p}(k)$ can then be used as an initial estimate for the quasi-Newton algorithm to predict the motion between the frames I_n and I_{n-k-1}, represented by the coefficient vector $\boldsymbol{p}(k+1)$. Thus the evolution of the shapes of the objects throughout the time interval $(n, n-\tau)$ are inherently modeled by a tubular surface where the shape of the object at each discrete time instant corresponds to a cross-section of the surface, as illustrated in Fig. 4.

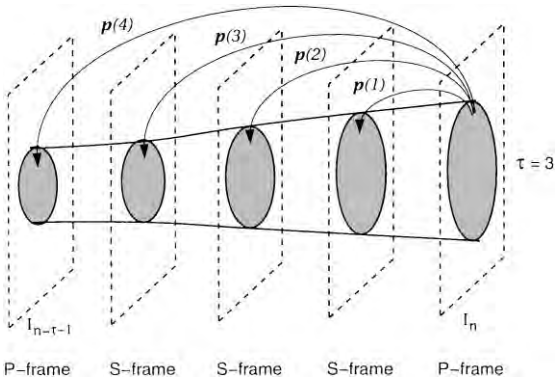

Fig. 4. Incorporation of the motion information in the skipped (S) frames. Notice that the motion estimation proceeds backwards.

Denoting the quadratic coefficient vector $\boldsymbol{p}(m,k)$ as a function of the resolution level m and the delay k, the combined multiscale procedure, which is used in our implementations, is given in the pseudo-code as follows:

initialize $\boldsymbol{p}(m_{\max}+1,0)$ (starting point)
for $k=1:\tau$ *do*
  for $m=m_{\max}:1$ *do*
    $I=h_m(x,y)*I_n(x,y)$
    $I'=h_m(x,y)*I_{n-k}(x,y)$
    set $\boldsymbol{p}_0(m,k)=\boldsymbol{p}(m+1,k-1)$
    minimize $E(\boldsymbol{p})$ for $\boldsymbol{p}(m,k)$ between I and I'
  end
end

where $h_m(x,y)$ is spatial averaging filter of size $m\times m$, $m_{\max}$ is the number of resolution levels, and $\boldsymbol{p}_0(m,k)$ denotes the initial value used for the quasi-Newton estimation of $\boldsymbol{p}(m,k)$. Thus the motion estimation evolves in a double loop, both from lower to higher resolution, and temporally from present frame τ past one.

Beside the improvement obtained in the estimation performance, the temporal evolution of the shapes of the objects between S-frames, as illustrated in Fig. 4, may lead to additional advantages. First, the temporal resolution can be increased at a small additional cost by predicting the skipped frames simply using τ extra motion information. This also results in segmented objects that are more attuned to the real objects of the scene, and the prediction of the object shapes in the skipped frames via the updated motion parameters can be utilized for object tracking purposes.

In conclusion, the multiscale approach makes the motion estimation more robust in that the complex algorithm described in Section 2.2 becomes more stable and noise insensitive. We should also point out that, although in our coding scheme, backward motion estimation is considered. However, backward and forward estimations are dual problems so that the same analysis is mostly valid for forward estimation. The forward estimation is more advantageous for temporal object tracking problem, but it is less robust because of the resulting ambiguous and conflicting image pixels, and the error accumulation through the reconstruction of consecutive image frames. On the other hand, the backward estimation has the advantage that object pixels are guaranteed to be assigned to their corresponding pixels in the previous frame. Moreover, the interpolation problem due to mapping of discrete image grid to a continuous plane can be solved reasonably well by the bilinear interpolation as in (11).

3. Motion segmentation

The goal of segmentation in an object-oriented coding scheme is to subdivide the scene into objects which are defined by the three parameter sets corresponding to the motion, shape and color (luminance as well as the chrominance) information. A segmentation process, which relies purely on motion information such as ours and [14], does not necessarily result in exact physical objects. The resulting regions reflect rather the 3D geometric structure and the temporal behaviour of the objects, as functions of the chosen object model, and may correspond in any one frame to differently moving parts of the real objects. These regions are called model compliance regions, whereas the regions for which the model fails are referred to as model failure regions. The latter may originate from covered, uncovered, or deformed regions or objects newly moving into the scene. In summary, the three object types are delineated in terms of their motion behaviour.

The final output of the motion segmentation scheme is a disjoint partition $\mathscr{P}$ of the whole image plane $\mathscr{R}$ in terms of background, model compliance and model failure regions which are defined as follows:

A region R_i is said to be a background region, and denoted by R_i^{BG}, if and only if the frame difference $\mathrm{FD}(R_i) \leqslant T^{\mathrm{BG}}$, where

$$\mathrm{FD}(R_i) = \frac{1}{N_i} \sum_{\boldsymbol{x} \in R_i} (I_n(\boldsymbol{x}) - I_{n-1}(\boldsymbol{x}))^2,$$

and T_{BG} is a predefined error threshold. The whole background region $R^{\mathrm{BG}} = \bigcup_{i=1}^{\mathcal{N}_{\mathrm{BG}}} R_i^{\mathrm{BG}}$, where $\mathcal{N}_{\mathrm{BG}}$ is the number of background regions, consists of those patches where the temporal changes are not significant and correspond to either stationary background or the objects with negligible motion. A region $R_i \subset \mathscr{R} - R^{\mathrm{BG}}$ is said to be a model compliance region, denoted by R_i^{MC}, if and only if $\min_A E(R_i, A) \leqslant T_{\mathrm{MC}}$, where T_{MC} is the error threshold determining whether the motion model fails or not. A region $R_i \subset \mathscr{R}$ is said to be a model failure region, denoted by R_i^{MF}, if and only if $\min_A E(R_i, A) > T_{\mathrm{MC}}$, and consequently, no parameter set A_i can be associated with such a model failure region. In conclusion, the whole partition is given by

$$\mathscr{R} = R^{\mathrm{BG}} \cup R^{\mathrm{MC}} \cup R^{\mathrm{MF}}, \tag{15}$$

where $R^{\mathrm{MC}} = \bigcup_{i=1}^{\mathcal{N}_{\mathrm{MC}}} R_i^{\mathrm{MC}}$ and $R^{\mathrm{MF}} = \bigcup_{i=1}^{\mathcal{N}_{\mathrm{MF}}} R_i^{\mathrm{MF}}$. The total number of MC and MF regions are denoted by $\mathcal{N}_{\mathrm{MC}}$ and $\mathcal{N}_{\mathrm{MF}}$, respectively.

In an object-oriented coding scheme a region with its motion, shape and color properties is called an object. Model compliance and model failure objects are defined below: A model compliance object O_i^{MC}, associated with a model compliance region R_i^{MC}, is defined by two parameter sets, S_i^{MC} for the boundary shape of the support region and A_i^{MC} for the motion information. A model failure object O_i^{MF}, associated with a model failure region R_i^{MF}, is defined by two parameter sets, S_i^{MF} for the boundary shape of the support region and C_i^{MF} for the color information. Thus, the motion and the shape of MC objects, and the shape and the color of MF objects are coded and transmitted.

Finally, the segmentation information is described by a labeling defined as follows: The labeling map $\boldsymbol{L}: \Re^2 \to \Re$ associates each region R_i with a label such that

$$\boldsymbol{L}(x, y) = \begin{cases} 0 & \text{if } (x, y) \in R^{\mathrm{BG}}, \\ -i & \text{if } (x, y) \in R_i^{\mathrm{MF}}, \\ i & \text{if } (x, y) \in R_i^{\mathrm{MC}}. \end{cases}$$

3.1. Segmentation as an optimization

An object-oriented coder should optimize the tradeoff between cost and quality of coding. Thus both the modeling error in MC regions, the coding cost of the partition information and the size of resulting MF regions should be kept as small as possible. The overall segmentation-estimation problem can be formulated by the following constrained optimization problem:

$$\min_{\mathscr{P}} J + \lambda G \quad \text{subject to } \forall i,\ E(R_i^{\mathrm{MC}}, A_i^{\mathrm{MC}}) \leqslant E_{\max}, \tag{16}$$

where

$$J(\mathscr{R}^{\mathrm{MC}}, \mathscr{A}^{\mathrm{MC}}) = \sum_{i=1}^{\mathcal{N}_{\mathrm{MC}}} E(R_i^{\mathrm{MC}}, A_i^{\mathrm{MC}}) \tag{17}$$

and

$$G = G(\mathscr{S}^{\mathrm{MC}}, \mathscr{S}^{\mathrm{MF}}, \mathscr{A}^{\mathrm{MC}}, \mathscr{C}^{\mathrm{MF}}). \tag{18}$$

This optimization problem then becomes the problem of finding a partition $\mathscr{P}$ of $\mathscr{R}$ such that the function $J + \lambda G$ is minimized under the given constraint, where λ is a weighting constant. Here, J denotes the modeling error which is given by the model compliance regions and the associated motion parameters. In (17), $\mathscr{A}^{\mathrm{MC}} = \bigcup_{i=1}^{\mathscr{N}_{\mathrm{MC}}} A_i^{\mathrm{MC}}$ denotes the concatenation of the object motion vectors.

The coding cost, or equivalently the bit-rate G in (18) depends on the total shape parameter set $\mathscr{S}$ of the MF and MC regions, the whole motion parameter set of the MC regions and the color of the MF regions. Recalling that the cost of color coding is relatively high, the size of the MF regions and the number of the resulting MC regions should be kept as small as the resulting modeling error in (17) allows. Moreover, smooth boundary contours are desired, since the shape coding cost (18) also increases with the complexity and details of the contours. Finally, note that (17) assumes errorless parameter coding, i.e., the error resulting from parameter coding is not included in the objective function.

The solution of this sizeable nonlinear optimization problem, which can also be considered as an optimum description problem, is only possible via iterative procedures in which the segmentation and estimation processes are employed in interaction. Although such iterative approaches to this optimization problem can only yield suboptimal solutions, in practice, one obtains adequate performance.

3.2. A novel motion segmentation

The segmentation methods in object-oriented schemes, using motion information, are usually hierarchically structured [14,4], that is proceeding from larger objects to smaller ones. The hierarchy proceeds from the gross temporally changed and unchanged regions of the scene, to increasingly finer differentiation of changed regions using motion compliance. This hierarchical procedure is iterated until all the objects are differentiated so as to satisfy the object model, including all differently moving parts of the physical objects, the objects under occlusion or the objects occluding larger moving objects. This hierarchical procedure can be visualized in Fig. 5.

In the spatio-temporal segmentation schemes such as [19], the scene is first oversegmented into spatially homogeneous atomic regions. These atomic regions are then successively merged using the motion information, though the concepts of model compliance and model failure are not used. The segmentation scheme, thus, can be regarded as proceeding from smaller objects to larger ones.

The segmentation scheme proposed in this paper, is based on pure motion information keeping the framework of model compliance and model failure regions of [14]. However, the object hierarchy is reverted so as to proceed from smaller objects to larger ones, thus allowing a more flexible segmentation which is closer to the optimum partition and which is applicable to a more general class of image sequences. The bottom-up scheme is superior to the top-down approach in that (a) object detail can be controlled at any resolution level, (b) the chance of missing small but semantically relevant objects, such as eyes and mouth in videophony is minimized, and (c) the scheme is independent of the scene content. On the contrary, the top-down scheme can miss semantically important details and it is too much oriented towards head-and-shoulder scenes. The proposed scheme is based on region growing and region merges with re-estimation of the motion parameters after a joint operation.

In brief, the proposed segmentation technique, which employs a pixelwise region growing process, relies purely on motion information so that no initial spatial segmentation is needed. The segmentation scheme as illustrated in Fig. 6 consists of six subtasks, two initial blocks for change detection and splitting into seed blocks, three blocks in the loop for motion estimation, merging and region growing, respectively, and one final block for eliminating spurious details. In the sequel these subtasks are detailed:

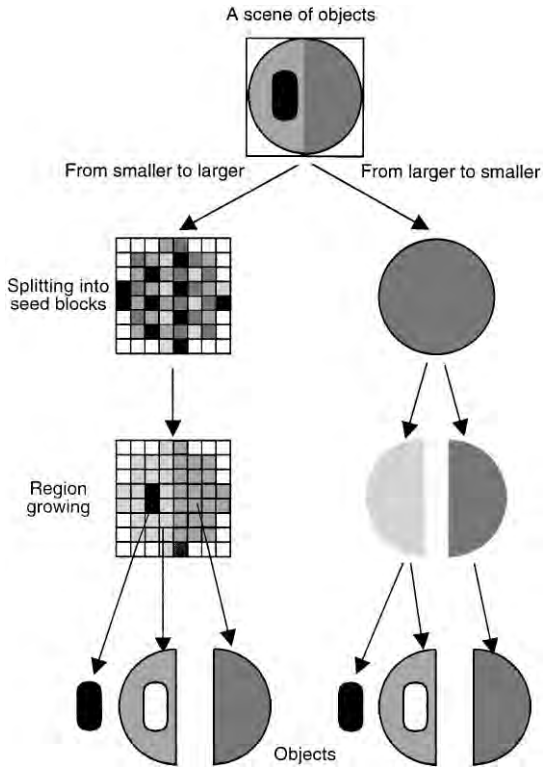

Fig. 5. Illustration of the object hierarchy in an object-oriented coding scheme.

3.2.1. Change detection map

The change detection subtask aims to differentiate the temporally changed and unchanged regions of the scene. Unchanged regions denoted by $\mathscr{R}^{BG}$ correspond to either stationary background or the objects with no apperant motion. In the change detection algorithm the image is first thresholded using the absolute difference of the low-pass filtered (3×3) versions $\tilde{I}_n(x, y)$ and $\tilde{I}_{n-1}(x, y)$ of consecutive images. Thus any FD below a threshold T_{CH} is labeled as background ($\boldsymbol{L}(x_k, y_k) = 0$), any above is initially declared as model failure ($\boldsymbol{L}(x_k, y_k) = -1$). The label map is further subjected to (3×3) median filtering in order to smooth the boundaries between the changed and unchanged regions, and to remove some of the isolated pixels. In the binary map $(0, -1)$ any small region whose size is smaller than a predefined threshold, which was not removed by median filtering, is assigned to its surrounding region label.

3.2.2. Seed blocks

The region growing process starts from seed blocks that have motion model error $E(R_i^{MC}, A_i^{MC}) \leqslant T_{MC}$. Thus the changed regions (regions with label "-1") are split into non-overlapping blocks B_i of constant size (e.g. 7×7), each subjected to a motion estimation process in terms of the quadratic model parameters. If a block B_i partly consists of unchanged pixels, then the estimation is carried out only for the part corresponding to the changed pixels. Those blocks for which the model error is below the predefined threshold T_{MC} are labeled as the seed blocks of the objects in the scene. The seed blocks are assumed to fall inside the object regions and prone to further growth. The blocks for which the model error is above the

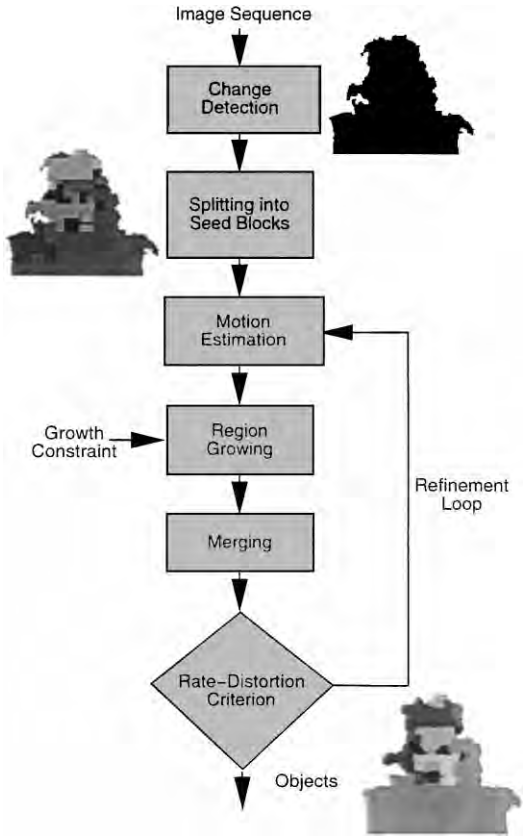

Fig. 6. Flowchart of the motion-based region growing segmentation.

threshold T_{MC} are assumed to fall either onto the boundaries of differently moving objects or into the model failure regions. These boundary blocks are not assigned any label and are handled in the following steps.

The size of the blocks in the initial partition should be chosen large enough such that the number of pixels in the blocks suffices for an accurate estimation of the 12 quadratic transform coefficients, but small enough to delineate the objects of interest in the scene. So the algorithm is as follows:

for each block B_i *do*
  $E^* = \min_{A_i} E(B_i, A_i)$
  if $E^* \leqslant T_{\mathrm{MC}}$ *then* $\boldsymbol{L}(x, y) = i \ \forall (x, y) \in B_i$
end

The output of this algorithm, as illustrated in the second silhouette in Fig. 6, is a rough segmentation of the scene into MC regions, MF regions and background, which needs to be further refined.

3.2.3. Region growing

To refine the rough partition resulting from "seed blocks" stage, a merging and pixelwise growing process of the model compliance regions into both the neighboring model failure and model compliance regions, is executed.

Region growing of MC regions into MF regions. Since the color information of MF objects is expensive to code, these regions should be minimized in both size and number. The initial roughly segmented MF regions may partly contain MC pixels, either because a block B_i may sit astride an MF and a neighboring MC region, or astride two neighboring MC regions, each having different motion vectors, for which a unique motion cannot be determined. To this effect in order to reduce the MF pixels, neighboring MC regions are grown into MF regions to identify MF pixels that would rightly belong to a MC region according to the associated motion. The resulting algorithm is as follows:

for each MF pixel (x, y) neighboring to R_i^{MC} *do*
  if $E((x, y), A_i^{\mathrm{MC}}) \leqslant T_{\mathrm{MC}}$ *then* $\boldsymbol{L}(x, y) = i$
end

The above process is iterated until there remains no MF pixel (x, y) such that $E((x, y), A_i^{\mathrm{MC}}) \leqslant T_{\mathrm{MC}}$, for any neighboring R_i^{MC}.

Region growing of a MC region into another MC region. Neighboring MC regions are also grown into each other for those pixels of region i that would be better represented by the motion model of region j. The pixels at the boundary of a region R_i^{MC} are tested for the motion parameters A_j^{MC} associated with the neighboring region R_j^{MC}, and provided that the resulting error is less than that of A_i^{MC}, the pixel is assigned to the neighboring region. Then the following procedure is iterated:

for each pixel $(x, y) \in R_i^{\mathrm{MC}}$ neighboring to R_j^{MC} *do*
  if $E((x, y), A_j^{\mathrm{MC}}) < E((x, y), A_i^{\mathrm{MC}})$ *then* $\boldsymbol{L}(x, y) = j$
end

Region growing of a MF region into a MC region. The MC regions of the initial segmentation may also involve MF pixels. Thus a region growing process should be employed so that MF regions grow into MC regions by checking the modeling error. The iteration of the following procedure aims to exploit these MF pixels:

for each MC pixel $(x, y) \in R_i^{\mathrm{MC}}$ neighboring to $\mathscr{R}_j^{\mathrm{MF}}$ *do*
  if $E((x, y), A_i^{\mathrm{MC}}) > T_{\mathrm{MC}}$ *then* $\boldsymbol{L}(x, y) = -j$
end

Refinement of the background regions. The threshold T_{CH} of the change detection algorithm is set to be smaller than the background discrimination threshold T_{BG} to prevent spurious background regions resulting within homogeneous gray level patches in the MC regions. Therefore, the boundary between the MC regions and the background should also be refined by the following algorithm:

for each MC pixel $(x, y) \in R_i^{\mathrm{MC}}$ neighboring to $\mathscr{R}^{\mathrm{BG}}$ *do*
  if $|I_n(x, y) - I_{n-1}(x, y)| \leqslant E((x, y), A_i^{\mathrm{MC}})$ *then* $\boldsymbol{L}(x, y) = 0$
end

and

for each BG pixel $(x, y) \in \mathscr{R}^{\mathrm{BG}}$ neighboring to R_i^{MC} *do*
  if $|I_n(x, y) - I_{n-1}(x, y)| > E((x, y), A_i^{\mathrm{MC}})$ *then* $\boldsymbol{L}(x, y) = i$
end

In addition, during the region growing process, each MC region R_i^{MC} for which the frame difference becomes less than the background discrimination threshold, that is $FD(R_i^{MC}) \leqslant T^{BG}$, should wholly be assigned to the background.

3.2.4. Refinement loop

As seen in Fig. 6, the initial segmentation scheme is refined in a loop of merging, region growing, and motion estimation blocks to attain an optimal partition, and hence the bit-rate versus quality tradeoffs. The refinement loop has the following characteristics:

1. Region growing process is based on the assumption that the motion parameter set estimated for a MC region is also valid for the outside pixels near its boundary. Thus the motion estimation procedure inherently estimates an object surface $f(X,Y,Z) = 0$, for which the motion parameter set is denoted by A_f. All yet unlabeled boundary pixels are also considered part of this surface so long $f(X_k,Y_k,Z_k) \approx 0$ is approximately satisfied, hence the same A_f is also valid for (X_k,Y_k,Z_k). Usually, as the regions grow away from their initial support region their surface structure also changes such that this approximation will eventually lose its validity. When a MC region cannot be further grown by the current parameter set A_f, i.e., by the current surface representation $f(X,Y,Z) = 0$, a new set of motion parameters, denoted by A_g, are estimated so that the resulting model is a better reflection of the grown object surface $g(X,Y,Z) = 0$. Therefore, at each cycle of the region growing loop, a re-estimation of the motion model parameters of the MC regions which have grown (or have become smaller) considerably (e.g., 10%) at the previous cycle is carried out.

2. The MC regions, which have newly been neighbors by having swallowed the intervening regions in the growth process, can potentially be merged at the successive cycles of the loop.

3. In the refinement loop the error threshold T_{MC} can gradually be increased as one cycles through the loop. With this gradual increase, the ambiguous MF pixels that are simultaneously contended by two different MC regions can be correctly attributed to the MC region yielding a lower compensation error. Gradual increase of T_{MC} can also be utilized for adjusting the cost-quality tradeoff, that is the error threshold is increased gradually until the total size of MF regions becomes sufficiently small.

3.2.5. Motion re-estimation

In order to avoid the computational load of the quasi-Newton minimization at each cycle, a strategy which is similar to the one used in [16] is adopted. The method presented in [16] is utilized for region merging process, whereas the method described below aims to re-estimate the motion parameters of each model compliance region by making use of its already estimated parameters in previous iterations.

Let $\tilde{R}_i^{MC}$ denote the updated version R_i^{MC} so that $(\tilde{R}_i^{MC} - R_i^{MC}) \cup (R_i^{MC} - \tilde{R}_i^{MC})$ corresponds to the pixels which are removed or added by the motion field extension. The problem is to minimize the mean square error function given by

$$\sum_{(x_k,y_k)\in\tilde{R}_i^{MC}} (I_n(x_k,y_k) - I_{n-1}(\tilde{u}_k,\tilde{v}_k))^2 \tag{19}$$

and to obtain a new parameter vector $\tilde{\boldsymbol{p}} = [\tilde{\boldsymbol{a}},\tilde{\boldsymbol{b}}]$ by using the already estimated $\boldsymbol{p}$. The trick leading to a fast solution is that u_k and v_k are already sufficiently good estimates, which are given by

$$u_k = \boldsymbol{a}^T\boldsymbol{q}_k, \qquad v_k = \boldsymbol{b}^T\boldsymbol{q}_k \tag{20}$$

for all $(x_k,y_k)\in\tilde{R}_i^{MC}$, where $\boldsymbol{q}_k = [x_k^2, y_k^2, x_k y_k, x_k, y_k, 1]$.

The Taylor series expansion of $I_{n-1}(\tilde{u}_k, \tilde{v}_k)$ around (u_k, v_k) gives

$$I_{n-1}(\tilde{u}_k, \tilde{v}_k) = I_{n-1}(u_k, v_k) + \frac{\partial I_{n-1}(u_k, v_k)}{\partial x}(\tilde{u}_k - u_k) + \frac{\partial I_{n-1}(u_k, v_k)}{\partial y}(\tilde{v}_k - v_k), \tag{21}$$

where the second and higher order terms can be neglected since the initial estimates u_k and v_k are sufficiently good. The above linear approximation, together with (19) yields the following equation for each $(x_k,y_k) \in \tilde{R}_i^{\mathrm{MC}}$:

$$\frac{\partial I_{n-1}(u_k, v_k)}{\partial x}\tilde{u}_k + \frac{\partial I_{n-1}(u_k, v_k)}{\partial y}\tilde{v}_k = I_n(x_k, y_k) - I_{n-1}(u_k, v_k) + \frac{\partial I_{n-1}(u_k, v_k)}{\partial x}u_k + \frac{\partial I_{n-1}(u_k, v_k)}{\partial y}v_k. \tag{22}$$

Overall, this results in an overdetermined system of linear equations:

$$\boldsymbol{Q}\boldsymbol{p} = \boldsymbol{z}, \tag{23}$$

where the kth rows of $\boldsymbol{Q}$ and $\boldsymbol{z}$ are given by

$$\left[\frac{\partial I_{n-1}(u_k, v_k)}{\partial x} \quad \frac{\partial I_{n-1}(u_k, v_k)}{\partial y}\right] \otimes [1 \quad x_k \quad y_k \quad x_k y_k \quad x_k^2 \quad y_k^2] \tag{24}$$

and

$$\boldsymbol{z}(k) = I_n(x_k, y_k) - I_{n-1}(u_k, v_k) + \frac{\partial I_{n-1}(u_k, v_k)}{\partial x}u_k + \frac{\partial I_{n-1}(u_k, v_k)}{\partial y}v_k. \tag{25}$$

Solution of (23) in terms of $\boldsymbol{a}$ and $\boldsymbol{b}$, thus the new parameter set $\tilde{A}$ associated to $\tilde{R}_i^{\mathrm{MC}}$ can be obtained by the well-known least-squares method.

3.2.6. Merging

As far as the flexibility of the object model allows, the number of the MC regions should be kept small to curb the coding cost of motion and shape parameters. Successive merging of neighboring MC regions for which the motions can be represented by a single set of motion parameters can be achieved as follows:

for each neighboring R_i^{MC} and R_j^{MC} *do*
  $E^* = \min_A E(R_i^{\mathrm{MC}} \cup R_j^{\mathrm{MC}}, A)$
  if $E^* \leqslant T_{\mathrm{MC}}$ *then do*
    $\boldsymbol{L}(x, y) = i,\ \forall (x, y) \in R_j^{\mathrm{MC}}$
    $A_i^{\mathrm{MC}} = A^*$
  end
end

The above merging algorithm should be iterated so that the model compliance regions grow gradually by merging until no neighboring MC regions with similar motion are left. The motion re-estimation for A^* is again carried out as in the previous step. The minimization strategy for merging is based on the fact that the motion of the regions to be merged is already known from the previous steps of the segmentation algorithm. Consider two neighboring MC regions R_i^{MC} and R_j^{MC}, and the associated quadratic transform coefficient vectors $\boldsymbol{p}_i = [\boldsymbol{a}_i, \boldsymbol{b}_i]$, $\boldsymbol{p}_j = [\boldsymbol{a}_j, \boldsymbol{b}_j]$. The problem is the determination $\boldsymbol{p}^*$ such that

$$\sum_{(x_k, y_k) \in R_i^{\mathrm{MC}} \cup R_j^{\mathrm{MC}}} (I_n(x_k, y_k) - I_{n-1}(u_k^*, v_k^*))^2 \tag{26}$$

by using the already known u_k and v_k. When expanded by Taylor series around the already known (u_k, v_k) for each $(x_k, y_k) \in R_i^{MC} \cup R_j^{MC}$, which are sufficiently good estimates, the dependence of $I_{n-1}(u_k^*, v_k^*)$ on $\tilde{u}_k - u_k$ and $\tilde{v}_k - v_k$ becomes linear and the solution can be obtained by the least squares solution. This scheme for region merging turns out to be better than, for example, motion field extension [28] whereby the motion parameters for the whole merged region are equated to the motion parameters of the larger of the two regions. Another alternative is to write a linear system of equations which is given by

$$u_k^* = u_k, \qquad v_k^* = v_k \tag{27}$$

for all $(x_k, y_k) \in R_i^{MC} \cup R_j^{MC}$ and to solve for $\boldsymbol{p}^*$. One can make recourse to these latter schemes, as in our implementation, only when the merging method which is based on the Taylor series expansion as expounded in (23), fails.

3.2.7. Post-processing of the segmentation

A growth constraint is imposed to region growing of the refinement loop, that demands that every acquired pixel must have at least three other neighbors of the same label. However, despite this growth constraint, the region boundaries can still be too active, and some post-processing of region contours is needed to eliminate the spurious details of the segmentation output, corresponding to small regions or regions with noisy boundaries, which unnecessarily increase the coding cost. Region boundaries are smoothed as follows:

- *Majority filtering.* In a 3×3 mask, the label of a pixel is changed to the label most frequently occurring.
- *Morphological closing* with a disk structuring element of a small radius [11], e.g. $r = 1$, smooths the boundaries, fuses narrow breaks and long thin gulfs, eliminates small holes, and fills gaps on the boundaries. Closing operation is applied successively to each region in the segmentation map. The order of the operation is important for the accuracy of the segmentation, since the closing operation has the effect of enlarging the regions: First, the MF regions should be closed, then the MC regions and finally the background regions.
- *Eliminating the small regions*. The regions, either MF, MC or background, which are smaller than a predefined threshold size, are eliminated by merging. The MF and background regions smaller than the threshold are merged into the neighboring larger regions with the most similar motion. The small MC regions are treated differently since they may still involve parts belonging to different larger MC regions. Thus, these are set to be unlabeled and consequently a region growing loop is restarted for those regions.

3.3. Discussion of motion segmentation

The proposed purely temporal segmentation process stands somewhere in between spatio-temporal schemes and the top-down temporal proposed in [19] and [14], respectively.

There are several shortcomings of the top-down algorithm in [14]. The first one is due to the hypothesis that the objects corresponding to higher steps of hierarchy dominate those of lower hierarchy in size. This hypothesis may cause problems since differently moving but connected regions of the scene do not necessarily have to dominate each other. In this case of differently moving but connected regions not dominating each other, the motion estimation process may result in inaccurate results. Second, even in the case the objects at different hierarchy levels really dominate each other, inaccurate results may still occur; sharp temporal changes may indeed affect the performance of the motion estimation process, regardless of the size of the objects. In addition, such a hierarchical scheme is capable of differentiating only the regions

with strongly diverse motions, and usually results in too few MC regions. For example, in a videophone scene, the head and shoulder are together treated as a single object and the deformable features such as eyes, and mouth as MF objects. Such a hierarchical structure proceeding from larger objects to smaller ones will in general lead a solution for (16) far from the optimum, and is constrained since it is applicable to only head-shoulder scenes.

On the other hand, spatio-temporal schemes rely on spatially homogeneous atomic regions. These atomic regions demand very accurate initial spatial segmentation since no further refinement is done. The spatial segmentation, as in [19] for example, utilizes very sophisticated and powerful morphological methods. However, a segmentation which does not properly take into account the temporal information can potentially conflict with the real motion in the scene. An example of this phenomenon is the case of a spatially homogeneous region containing differently moving objects. Indeed, the coding schemes of [19,27] do not much rely on the estimated motion parameters for the synthesis, but more on the color (texture) coding of the homogeneous object regions.

Our motion-based region growing scheme, which proceeds from smaller objects to larger ones, overcomes the above mentioned shortcomings of both spatial segmentation schemes and motion-based top-down schemes.

4. Parameter coding

The motion and shape of MC regions, and the shape and color of MF regions need to be coded efficiently. The shape of the background region needs not to be considered since it is simply the complement of the union of MC and MF regions: $\mathscr{R}^{\mathrm{BG}} = \overline{\mathscr{R}^{\mathrm{MC}} \cup \mathscr{R}^{\mathrm{MF}}} = R - (\mathscr{R}^{\mathrm{MC}} \cup \mathscr{R}^{\mathrm{MF}})$. Recall that MC (model compliance) regions are defined so that their motion can be represented well enough by the motion model under the chosen error criteria. Thus, motion model error residues need not to be coded neither.

4.1. Shape polygonization

The shape of an object can be determined by its boundary contour, and the corresponding segment label information can be retrieved by filling inside the closed contour. Chain codes [8,6], spline approximation [13] and polygonization [18,30] are the most common methods utilized for contour coding in object-oriented coding. Chain codes are usually used for lossless or very accurate representation purposes whereas the methods based on polygonization and spline representation yield only approximate solutions. However, the coding cost of the latter techniques are much less compared to chain coding.

We present here a novel shape coding strategy which is based on an approach in [31]. The boundary contour is approximated by polygon vertices, which are then coded differentially. Such a coding scheme inherently performs smoothing on the original contour and reduces significantly the coding cost.

Despite the smoothing actions on the contour as detailed in Section 3.2, the resulting contours can still profit from the simplification and smoothing of polygonization. However, the tradeoff between the coding cost and quality should carefully be taken into account.

4.1.1. Formulation of the problem

An original boundary is an ordered set of connected pixels, which can be denoted by $S = \{s_1, \ldots, s_l, \ldots, s_{N_S}\}$, where N_S is the number of the pixels on the boundary. The goal is to approximate the boundary by an ordered set of vertices, $V = \{v_1, \ldots, v_k, \ldots, v_{N_V}\}$, where N_V is the number of vertices. The vertices v_k are coded differentially in terms of the successive differences of the vertex coordinates. Thus the total bit-rate (number of bits) Γ is the sum of the individual bit-rates $\gamma(v_{k-1}, v_k)$ corresponding to

consecutive bit expenditures and can be written as

$$\Gamma(v_1, \ldots, v_{N_v}) = \sum_{k=1}^{N_V} \gamma(v_{k-1}, v_k), \tag{28}$$

where $\gamma(v_0, v_1)$ is set to zero.

Each couple of consecutive vertices v_{k-1} and v_k defines a line segment, i.e., an edge of the polygon representation. Let $P(v_{k-1}, v_k)$ be the ordered set of the resulting connected pixels of the polygon edge defined by v_{k-1} and v_k such that $P(v_{k-1}, v_k) = \{p_1 = v_{k-1}, p_2, \ldots, p_j, \ldots, p_{N_P^k} = v_k\}$, where N_P^k denotes the number of the pixels in the kth edge of the polygon. Then the set of all polygon pixels becomes $P = \bigcup_{k=1}^{N_V} P(v_{k-1}, v_k)$. Polygon approximation causes geometric shape distortion, which can be effectively quantified by the Hausdorff metric, i.e. the absolute distance between S and P_k such that

$$D_k(v_{k-1}, v_k) = \max\left\{ \max_{p_i \in P(v_{k-1}, v_k)} \min_{s_j \in S} d(p_i, s_j), \max_{s_i \in S} \min_{p_j \in P(v_{k-1}, v_k)} d(s_i, p_j) \right\}, \tag{29}$$

where $d(p_i, s_j)$ is the Euclidean distance between the pixels p_i and s_j. Then the overall distortion is given by

$$D(v_1, \ldots, v_{N_v}) = \max_{k \in \{1, \ldots, N_V\}} D_k(v_{k-1}, v_k). \tag{30}$$

The goal of shape coding is to optimize the tradeoff between the bit-rate Γ and the shape distortion D, which can be formulated as a constrained optimization problem such that

$$\min_{v_1, \ldots, v_{N_V} \in \mathscr{R}} \Gamma(v_1, \ldots, v_{N_v}) \quad \text{subject to } D(v_1, \ldots, v_{N_v}) \leqslant D_{\max}. \tag{31}$$

The vertices $v_1, \ldots, v_{N_v}$ are preferably chosen to lie on the boundary such that $v_k \in S$ as in [30], or within a band of width $2D_{\max}$ as in [31]. The pixels in this band are ordered such that every pixel is associated to a boundary pixel s_l and considered to be a vertex candidate. The optimization problem (31) then becomes a shortest path problem for a directed acyclic graph and is solved via an efficient algorithm based on the observation that given a certain vertex of a polygon, the selection of the next vertex is independent of the selection of the previous vertices [30].

The main criticism of this formulation is that (31) defines a context-free optimization problem in the sense that it does not take into account the reconstruction error of the region. Recall that the goal is the minimization of the mean square synthesis error, while shape optimization as in [31,30] does not necessarily optimize the coder performance.

Our boundary shape coding algorithm takes into account both the geometrical distortion as well as synthesis error, as explained below. We start with the following definition.

Definition 4.1. Let R, S and T denote, respectively, a generic model compliance region, its boundary and the band of ordered sets $T(s_l)$. The subset $T(s_l)$ associated to a boundary pixel s_l is the set of ordered pixels t_l^i with coordinates $(x, y) \in \mathscr{R}$ if and only if one of the following two conditions is satisfied:

1. $(x, y) \in \bar{R}$, and

$$d((x, y), s_l) \leqslant D'_{\max},$$

$$E((x, y), A) \leqslant E_{\max},$$

there exists no $(x', y') \in \bar{R}$ such that

$$d((x', y'), s_l) \leqslant d((x, y), s_l) \text{ and } E((x', y'), A) > E_{\max},$$

2. $(x, y) \in R$ and $d((x, y), s_k) \leqslant D_{\max}$,

where $\bar{R}$ denotes the complement, i.e., the exterior of R and $E((x, y), A)$ is the synthesis (modeling) error at (x, y), and A is the motion parameter set associated to R. $E_{\max}$ is the tolerated synthesis error due to shape approximation, and $D'_{\max}$ and $D_{\max}$ are the maximum tolerated geometric shape distortions towards the outside and the inside of the boundary, respectively.

The construction of T is effected by sliding two circular disks with radii $D'_{\max}$ and $D_{\max}$ along the boundary and associating the pixels inside the disks with the boundary pixels as long as the constraints, in terms of the geometric distortion and synthesis error, are satisfied as shown in Fig. 7. Notice that for the second condition in Definition 4.1, the synthesis error requirement is automatically satisfied since the pixel is inside a model compliance region. Each $T(s_l)$ turns out to be the union of two partial circular disks with radii $\rho'(s_l) < D'_{\max}$ for the outer one, and $\rho(s_l) = D_{\max}$ for the inner. $D'_{\max}$ is chosen as much larger than $D_{\max}$ so that the sets $T(s_l)$ form a band which is wider in the outside of R. Due to Definition 4.1, the outer part of each $T(s_l)$ becomes the outer sector of the maximal circular disk within which the motion parameter A complies. These sectors are illustrated in Fig. 7.

The shape coding problem can then be reformulated as follows:

$$\min_{(v_1, \ldots, v_{N_v}) \in T} \Gamma(v_1, \ldots, v_{N_v}) \tag{32}$$

subject to

$$D_{\text{out}}(v_1, \ldots, v_{N_v}) \leqslant D'_{\max}, \qquad D_{\text{in}}(v_1, \ldots, v_{N_v}) \leqslant D_{\max}, \qquad E(v_1, \ldots, v_{N_v}) \leqslant E_{\max},$$

where D_{out} and D_{in} denote the shape distortion corresponding to parts of the polygon which are outside and inside of the original boundary, respectively. Note that the vertices are such that $(v_1, \ldots, v_{N_v}) \in T$ and $T(s_l)$'s are not disjoint so that they may have common pixels, unlike the band defined in [31]. In (32), $E(v_1, \ldots, v_{N_v})$ is the maximum synthesis error due to the polygonization, and given by

$$E(v_1, \ldots, v_{N_v}) = \max_{(x,y) \in R_P - R} E((x, y), A), \tag{33}$$

where R_P denotes the region inside the polygon, and R the region of the original shape. The optimum polygon tends mostly to widen towards the outside, allowing only a small distortion inside, and the difference $R_P - R$ corresponds to these extra pixels.

The advantage of this formulation compared to that in (31) [31,30] is that the geometric distortion constraint is more relaxed towards outside so that $D'_{\max}$ is chosen quite larger as compared to $D_{\max}$. Secondly the polygonization in (32) takes explicitly into account the synthesis error.

The construction of the band for model failure regions is carried out in a similar way, though, of course, an associated motion parameter A does not exist. Since the color coding of the model failure regions is expensive, augmentation of these regions towards outside must be more controlled, hence a smaller value of

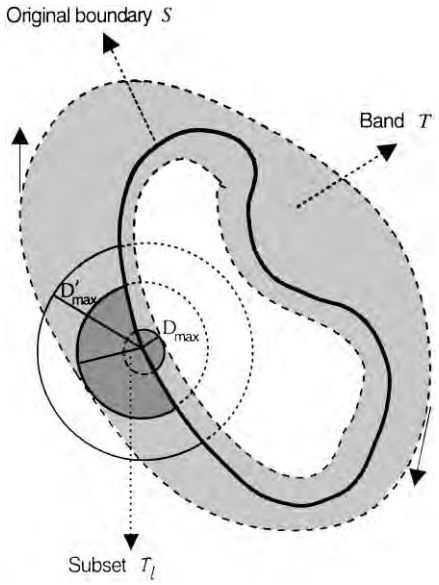

Fig. 7. Construction of the sensivity band for shape polygonization. The outer tolerance band has width controlled both by geometric distortion and by the motion compensation constraints ($\rho' \leqslant D'_{\max}$). The inner band is controlled only by geometric distortion ($D_{\max}$) since inside the motion compensation holds true.

$D'_{\max}$ is chosen than that in the case of model compliance regions. For example, $D'_{\max} = D_{\max}$ is one such option.

The subsets $T(s_l)$, i.e. the sets of vertex candidates, are ordered sets, which is required for a computationally efficient algorithm. When a vertex candidate is not lying on the boundary contour, we order the associated pixels t_l^i of the subset $T(s_l)$ to s_l in the order of increasing distance $d(t_l^i, s_l)$. Thus the ordering of the pixels in each $T(s_l)$ is such that if $i < j$ then $d(t_l^i, s_l) \leqslant d(t_l^j, s_l)$.

4.1.2. The coding algorithm

Since the optimum choice of a vertex is independent of the previously determined vertices, the coding algorithm can be defined edgewise. Let v_k^* be the optimum kth vertex associated to the boundary pixel denoted by $s_{v_k^*}$, and assume that v_{k-1}^* and its associated boundary pixel $s_{v_{k-1}^s}$ are also given. Recall that the relationship between any $(v_k^*, s_{v_k^*})$ pair is that of a polygonal vertex and the closest boundary pixel.

Definition 4.2. The regions $R_{\text{out}}(v_k)$ and $R_{\text{in}}(v_k)$ are defined as the set of pixels which are outside and inside of the region R, respectively, and which lie between the polygon edge $P(v_{k-1}, v_k)$ and S. These regions are bounded by the line segments $P(v_{k-1}, v_k)$, $P(s_{v_k}, v_k)$, $P(s_{v_{k-1}}, v_{k-1})$ and the curve determined by the set of boundary pixels between $s_{v_{k-1}}$ and s_{v_k}. $R(v_k)$ denotes $R_{\text{out}}(v_k) \cup R_{\text{in}}(v_k)$.

It follows from the above definition that $R_P - R = \bigcup_{k=1}^{N_V} R_{\text{out}}(v_k)$ and $R - R_P = \bigcup_{k=1}^{N_V} R_{\text{in}}(v_k)$, where $R(v_k)$'s are all disjoint (see Fig. 8).

Our aim is to end up with a computationally efficient shape coding algorithm which approximates the original noisy boundary S with a polygon P lying inside the band T as defined in Definition 4.1. It is also required that inside the approximate polygon there should be no holes, i.e., no pixels where the motion model

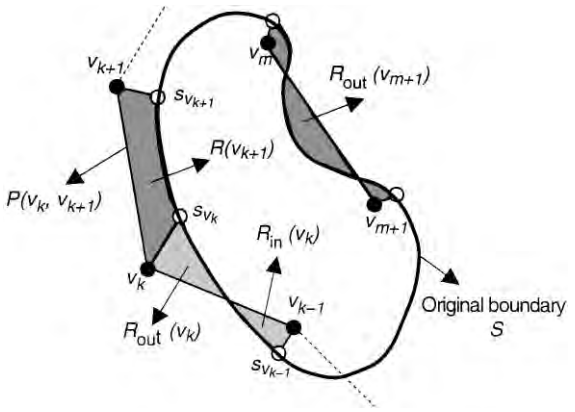

Fig. 8. Illustration of the difference between the polygon and the original boundary. Note that $R(v_{k+1}) = R_{out}(v_{k+1})$ since $R_{in}(v_{k+1})$ is missing.

fails. To define such an algorithm, below we introduce some propositions which are in fact geometrical consequences of the construction of the band T as proven in [34]. Therefore, one should keep in mind that the following observations hold for circular subsets $T(s_l)$, and are not necessarily valid for an arbitrary choice of the windowing shape, as in the case of, for example square windows.

Proposition 4.1. *If* $P(v_{k-1}, t_l^i) \cap T(s_m) \neq \emptyset \ \forall s_m$ *between* $s_{v_{k-1}}$ *and* s_l, *then* $P(v_{k-1}, t_l^i) \subset T$.

Thus, the polygon edge defined by the vertices v_{k-1} and t_l^i is assured to lie inside the band T if it has a common pixel with each vertex candidate set of the boundary pixels in between.

Proposition 4.2. *Each pixel in* $R(v_k = t_l^i)$ *satisfies the constraints of optimization in* (32), *if* $P(v_{k-1}, t_l^i) \cap T(s_m) \neq \emptyset \ \forall s_m$ *between* $s_{v_{k-1}}$ *and* s_l.

Proof. For any $(x, y) \in R(v_k = t_l^i)$ which is not in the edge set $P(v_{k-1}, v_k)$, consider the line perpendicular to the line of the edge $P(v_{k-1}, v_k)$. This line either cuts the partial shape contour or the line segments $P(s_{v_k}, v_k)$, $P(s_{v_{k-1}}, v_{k-1})$. If it cuts one of the line segments, say $P(s_{v_k}, v_k)$, then $d((x, y), s_{v_k}) < d(v_k, s_{v_k})$ so that $(x, y) \in T_l$. If it cuts the partial shape contour, then the intersection boundary pixel s_m on the partial shape contour satisfies $d((x, y), s_m) < d((x', y'), s_m)$, $\forall (x', y') \in P(v_{k-1}, t_l^i)$ so that by Proposition 4.1 and the given condition, $(x, y) \in T(s_m)$. Considering Definition 4.1, this implies $\forall (x, y) \in R(v(k) = t_l^i)$, $E((x, y), A) \leqslant E_{max}$ (if outside), and $d((x, y), s_m) < D'_{max}$.

The implication of this proposition is that once the band T has been constructed for the boundary of a region R, it suffices to check the polygon edge pixels and the related subsets for the given condition to assure that all the pixels in R_P satisfy the constraints of optimization.

Definition 4.3. An admissible vertex $t_l^i \in T$ satisfies the constraints of optimization, and $t_l^i \notin R(v_{k-1})$.

Recalling that the subsets of T are not disjoint, the requirement $t_l^i \notin R(v_{k-1})$ in the above definition is for keeping track of the original shape, as a precaution to the possibility of choosing the next optimum vertex v_k from the pixels in the region $R(v_{k-1})$.

Proposition 4.3. *If there exists no $t_l^i \in T(s_l)$ such that t_l^i is admissible then there exists no $t_m^i \in T(s_m)$ such that t_m^i is admissible for $m > l$.*

The above proposition stands as a stopping rule for the shape polygonization algorithm. Thus the algorithm does not necessarily visit all the boundary pixels, that is does not visit those remaining pixels that have no potential to become a vertex.

Argument 4.1. *If $s_l \in S$ is the furthest possible boundary pixel from $s_{v_{k-1}}$ for which a $t_l^i \in T(s_l)$ is admissible, then v_k^* which is nearly optimum, is in $T(s_l)$.*

When the admissible vertices are constrained so as to belong to the original boundary, i.e., $D'_{\max} = 0$ and $D_{\max} = 0$, the proof of the above argument is very straightforward. Since the vertices are coded differentially, the optimization problem turns out to be finding the shortest path between two boundary pixels under the imposed constraints. Thus the next optimum vertex is always the furthest possible boundary pixel, in view of the fact that the shortest path between two points is the line segment connecting them. However, when $D'_{\max} \gg 0$, though rarely, the polygon can unnecessarily fluctuate from the original shape towards outside in order to achieve the furthest s_l. Meanwhile the optimum vertex, which can only be found by exhaustive search which is computationally unfeasible, may indeed be in T_m such that $m < l$. This, in turn, may result in a higher bit-rate and in unnecessary distortions. A remedy for this undesired phenomenon is to constrain the admissible vertices more strictly, by extending Definition 4.3 as follows:

Definition 4.4. A pixel $t_l^i \in T$ is an admissible vertex if and only if it complies with Definition 4.3, and moreover $d(t_l^i, s_l) \leqslant D_{\max}$.

It follows from the above definition that the polygon vertices are forced to be more faithful to the original shape than the polygon edges which can be placed anywhere in the constructed band T. Recalling $D'_{\max} \gg D_{\max}$, the polygon vertices are chosen to be very close to the original boundary pixels s_l. In addition, the second requirement of admissibility in Definition 4.3 need not be checked when, for example, $D_{\max} = 1$ as in our implementations.

Argument 4.2. *If $t_l^i \in T(s_l)$ is admissible then $v_k^* \neq t_l^j \ \forall j > i$.*

The above argument is a consequence of the pixel ordering, that is, if $j > i$ then $d(t_l^j, s_l) \geqslant d(t_l^i, s_l)$. The optimum vertex should be as faithful, i.e., close, as possible to the original shape for the sake of minimizing both the bit-rate, i.e., the path length, and the shape distortion. A consequence of Argument 4.2 is also that there is no need to check all the vertex candidates in order to find the optimum vertex.

Let the initial vertex v_1 be a point on the boundary, say s_1, which is ideally chosen as to be the boundary pixel with the largest curvature, and let $N_{T(s_j)}$ denote the number of pixels in the subset $T(s_j)$. In view of the above discussion, given the vertex v_{k-1} associated to s_{l-1}, a nearly optimum vertex v_k can be found by the following algorithm:

```
set v_k = s_l;
for j = l to N_S;
  for i = 1 to N_{T(s_j)};
    if t_j^i is admissible      (by Definition 4.4)
      set v_k = t_j^i;
      break;      (By Proposition 4.2)
  if v_k has not been set at iteration j break;      (By Proposition 4.3)
```

Due to Proposition 4.3, the algorithm does not necessarily visit all the boundary pixels; if a boundary pixel s_j having no admissible vertex in $T(s_j)$ is encountered, the iteration breaks. Similarly, all the vertex candidates t_j^i associated to a boundary pixel s_j are not necessarily checked due to Proposition 4.2; if an admissible vertex is found in $T(s_j)$, ignoring the remaining pixels, the iteration breaks and the algorithm proceeds with the next boundary pixel s_{j+1}.

4.1.3. Summary of the coding algorithm

The proposed shape coding algorithm is actually straightforward although the description of its implementation seems to be complicated. The idea underlying the algorithm can be summarized as follows: Given an optimum current vertex v_{k-1}, the next optimum vertex v_k is the furthest possible pixel which is in the close neighborhood of the original shape, and which also satisfies the two constraints of the optimization. The first constraint is that the geometric shape distortion of the resulting polygon edge $P(v_{k-1}, v_k)$ should be less than predefined values. These error thresholds are generally chosen to be smaller for the distortion towards inside the region than that of the distortion towards outside, in order to avoid unlabeled ambiguous pixels. The second constraint relates to the resulting synthesis error, given by the motion parameters. In other words, an optimum vertex v_k is chosen such that the additional pixels imported into the region, denoted by $R_{\text{out}}(v_k)$ in Fig. 8, have to comply with the parameter A of the MC region within a synthesis error bound $E_{\max}$. Once the band T is constructed around the boundary shape as defined in Definition 4.1, by Proposition 4.2 it suffices to check whether each subset $T(s_l)$ associated to the boundary pixels s_l between $s_{v_{k-1}}$ and s_{v_k} intersects with the polygon edge $P(v_{k-1}, v_k)$, in order to assure that the resulting polygon edge satisfies the constraints of the optimization.

4.1.4. Vertex encoding

The vertices are coded differentially so that only the xy coordinate differences of the consecutive vertices are considered. A maximum difference of 16 is allowed for each coordinate; if the difference is larger than 16, then the vertex edge is partitioned into co-linear vertices. Moreover, due to the flexibility of placing the vertices within a neighborhood, they can be chosen at pixels with only even (or odd) coordinates. Consequently, each vertex coordinate can be differentially coded with 4 bits, i.e., each vertex costs of 8 bits totally.

Depending upon the value of $D_{\max}$ which determines the maximum tolerated shape distortion towards inside, the shape coding process may result in unlabeled pixels in the segmentation, that is pixels not belonging to any region R. These unlabeled pixels are assigned to the nearest R_i, which could be of model compliance, model failure or background type. Conversely, due to the shape approximation, some pixels which may end up belonging to more than one region. Such ambiguous pixels are handled as follows: If one of the multiple regions covering the ambiguous pixels is a model failure region, the pixel is assigned to this model failure region. If not, then the pixel is compensated by the associated motion parameter set of the model compliance region whose boundary is the furthest from that pixel. Another possibility is to compensate the pixel by taking the average of the individual displacement vectors resulting from the motion parameters of the overlapping MC regions.

Two instances of the shape coding algorithm are illustrated in Fig. 9. One can notice that the polygon approximation simplifies the original contour by covering most of the concavities. In Fig. 10, the performance of the shape coding process is illustrated for whole segmentation maps resulting from two different test sequences. The compression gain may differ mainly depending upon the complexity and the context of the original contour. The average bit expenditures have come out to be 0.87 and 0.96 bits per contour point for Miss America and Carphone sequences, respectively. These figures, however, drop to 0.74 and 0.82 bits per contour point if only the shape coding of model compliance regions is considered.

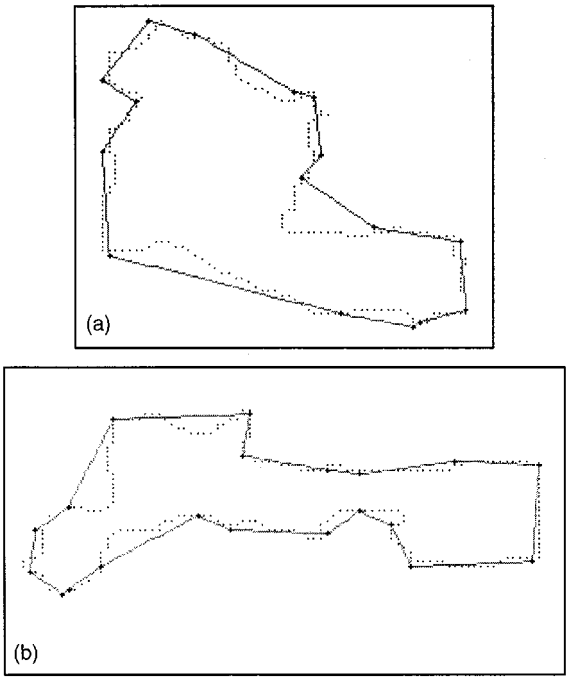

Fig. 9. Instances of the shape coding algorithm. The original contours of 220 and 241 pixels in the illustrations (a) and (b) are approximated by 16 and 20 vertices, respectively.

4.2. Color coding

Efficient coding of color of the MF objects is an important problem in object-oriented coding schemes. Classical block-based coding methods relying on DCT are not very convenient, since the resulting regions to be coded are arbitrarily shaped, and hence many blocks will have a hybrid support, that is partly MF and partly non-MF.

One solution to the coding of arbitrarily shaped regions is the region-based DCT transform coding, proposed by Gilge et al. [9]. The transform utilized in this technique is based on two-dimensional DCT base functions which are orthonormal with respect to the region of support. In [32], an approximate shape-adaptive technique has been proposed, which is based on predefined orthogonal base functions and does not require any orthogonalization, but in turn it is not statistically optimum. The advantage of this technique is its low computational complexity compared to the region-based DCT transform in [9].

Another solution to the problem of color coding in object-oriented schemes has been proposed in [29], which is based on a hybrid-adaptive DCT/DPCM scheme. The arbitrarily shaped region is split into blocks some of which are not complete, i.e., consist of pixels belonging to the outside of the region. The idea is that the sparse blocks which mostly consist of outside pixels can be coded more efficiently via DPCM coding. On the other hand, for the blocks which are almost complete, block-based DCT can be utilized to better exploit the spatial correlation.

Since the hybrid DCT/DPCM coding is more efficient compared to those in [9,32], we chose the hybrid DCT/DPCM method for our object-oriented coding scheme. Both DCT and DPCM coding techniques are applied to all blocks of each object separately for luminance and chrominance pixels. The method resulting in a shorter bit-stream is selected, assuring constant quality and fixed DCT and DPCM quantization step sizes for all the blocks of an object as in [29].

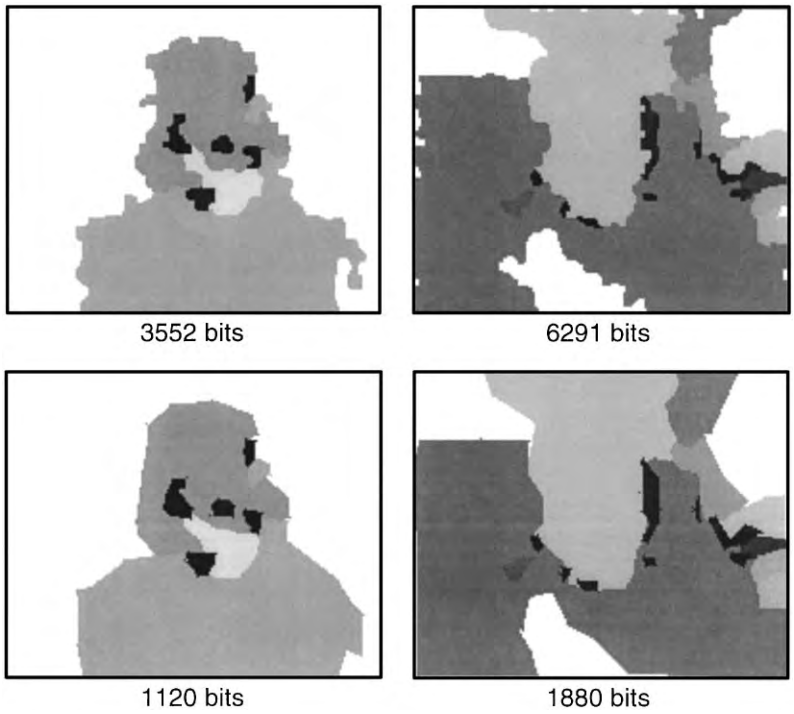

Fig. 10. Original and coded segmentation maps; Miss America (left), Carphone (right). In the above pair the bit expenditure for lossless chain coding is given, while in the figures below the bit total corresponds to the number of vertices $\times 8$.

4.3. Motion parameter coding

The motion of each object i is represented by a parameter set of $m = 12$ coefficients, $A_i = \{a_1, \ldots, a_6, b_1, \ldots, b_6\}$. However, the resulting coefficients have varying contributions in reducing the prediction (compensation) error, hence the coefficients with negligible contribution can be eliminated from the set A_i to obtain a reduced set of parameters $\tilde{A}_i$. Some of the base functions $\{x^2, y^2, xy, x, y, 1\}$ can be eliminated by checking the resulting additional prediction error. We use the coefficient removal procedure of [16], which is based on the elimination of the coefficients one by one, starting from those with higher degree terms. For each parameter set $\tilde{A}_i$ which is initially set to A_i, the algorithm is as follows:

1. Find $E_{\min} = E(R_i, \tilde{A}_i)$.
2. Remove a coefficient and its associated base function from the set $\tilde{A}_i$.
3. Minimize $E(R_i, \tilde{A}_i)$ and determine the minimum value $\tilde{E}_{\min}$, via the method used for region merging.
4. If $|E_{\min} - \tilde{E}_{\min}| < \varepsilon$, replace the removed coefficient and the associated base function.
5. Iterate steps 1, 2, 3 and 4 for each coefficient.

The motion estimation procedure of Section 2 results in real-valued coefficients which have to be quantized and converted into a bit-stream. Prior to the estimation, the coordinates of the bounding box of each region are normalized to $[-1, 1] \times [-1, 1]$ so that the coefficients A_i are all in the same range. Thus, a uniform quantizer can be designed adaptively, again by checking the resulting effect of the quantization on the prediction error. The region scaling factor need not be transmitted since it can be deduced from the shape information which is also available at the receiver.

The eliminated coefficients can be coded with a single flag bit. The nonzero coefficients which are concentrated around zero, are coded with eight bits, 7 bits for the quantizer and one bit for the sign flag. Three bits are spent to switch between eight different quantizer step sizes. The rest of the nonzero coefficients are coded by a code length of 15–20 bits.

The coefficient removal procedure not only reduces the bit budget, but also eliminates the error due to the quantization of small valued coefficients, since the procedure iteratively updates the remaining nonzero coefficients. The above-described motion parameter coding method yields nearly optimum experimental results which will be presented in the next section.

5. Experimental results

The performance of the proposed approach has been tested on the well-known test sequences Miss America, Claire, Carphone and Foreman in the QCIF format. Miss America and Claire sequences are typical video-telephony sequences with relatively small motion and stationary background, whereas Carphone and Foreman sequences contain more severe motion, moving background and tilting of the camera. Each sequence of 100 frames with the original frame rate 30 fr/s has been frameskipped by 6 so that the resulting sequence becomes sampled at the rate 5 fr/s. Though only the motion estimation-segmentation data is transmitted for every fifth frame, the motion information of the skipped frames has been also incorporated by the multiframe analysis to improve the estimation-segmentation performance.

The quality performance of the coding scheme is illustrated in Figs. 11–14 at different bit-rates. Bit-rate adjustment is obtained by tuning two main parameters of the segmentation scheme. The first parameter is the maximum allowed mean square error T_{MC} for the model compliance regions, which mainly determines the number of resulting MC regions and the size of model failure regions. The threshold T_{MC} has been chosen to vary in the range between 20 and 80 in our experiments. The MC regions have been coded only by their shape and motion parameters, neglecting the errors resulting from motion modeling. The second parameter is the constant quality required for color coding of the model failure regions, which varies between 33 and 36 dB. In all the experiments, the minimum allowed size in pixel count for the model failure has been set to 20, in order not to cause disturbing distortions in the small, but semantically important parts of the scene such as eyes and mouth, whereas the minimum size for the model compliance regions has been chosen so as to vary between 50 and 100.

Examples for the resulting segmentation maps, which illustrate the object differentiation capability of the scheme at different bit-rates, are displayed in Figs. 15–18, where the darkest regions, the white regions and the gray regions correspond to model failure, background and model compliance regions, respectively. These examples reflect the semantics of the scene satisfactorily. To give an idea or justify the region growing step in the motion segmentation algorithm the following pixel exchange traffic can be quoted. The average percentages of the exchanged pixels with respect to the QCIF size as a result of the region growing process in Section 3.2, have come out to be 5% between MC regions, and 7% between MC and MF regions for the Miss America sequence, whereas these percentages have been respectively 12% and 17% for the Carphone sequence.

Figs. 11 and 12 show that the quality of the coded sequences are quite good even at very low bit-rates for Miss America and Claire sequences, although there seem to exist small distortions at 16 kbit/s. The initial frames of Claire and Miss America sequences have been coded by intra-frame coding, however the overhead of intraframe coding is not included in the displayed results. On the other hand, the resulting bit-rates for Carphone and Foreman sequences are comparatively larger, and the quality of the coded frames deteriorates as the bit-rate decreases towards modem speeds. The evidence for this deterioration can be observed in the results given in Tables 1–4, where the total size of model failure regions comes out to be much smaller for Miss America and Claire sequences as compared to those sequences with more severe motion, with

Fig. 11. Coder performance for Miss America sequence at different bit-rates. In the order from top to down, original frames and coded frames at 37, 23 and 16 kbit/s, respectively.

Fig. 12. Coder performance for Claire sequence at different bit-rates. In the order from top to down, original frames and coded frames at 36, 28 and 17 kbit/s, respectively.

Fig. 13. Coder performance for Carphone sequence at different bit-rates. In the order from top to down, original frames and coded frames at 79 and 50 kbit/s, respectively.

proportional increases in the cost of color coding. Moreover, the number of MC regions for Foreman and Carphone sequences is a lot larger due to the nonstationary background, which in turn makes motion and shape coding more expensive.

Displayed segmentation maps in Figs. 17 and 18 show that there may exist many spurious model failure regions which are mainly due to sharp spatial changes where small displacement errors, which are not perceptually important, lead to very large synthesis errors. Thus, one can expect that elimination of these spurious MF regions, would cause significant decrease in the associated coding cost.

In order to illustrate the efficiency of our motion parameter coding scheme, the entropy of the motion coefficients $\tilde{A}_i$ (i.e., after pruning with zeros) and the resulting average codeword lengths for different sequences are given in Table 5. Our average codeword lengths are within 10% of the corresponding entropies, that is the motion parameter coding scheme is close to the optimum. It is also as expected but interesting to observe that the entropies of the motion parameters get larger as the motion in the scene becomes more severe. The ratio of the number of removed zero coefficients to the total number of coefficients is also included in Table 5. Let us point out that removal of coefficients corresponds to the simplification of

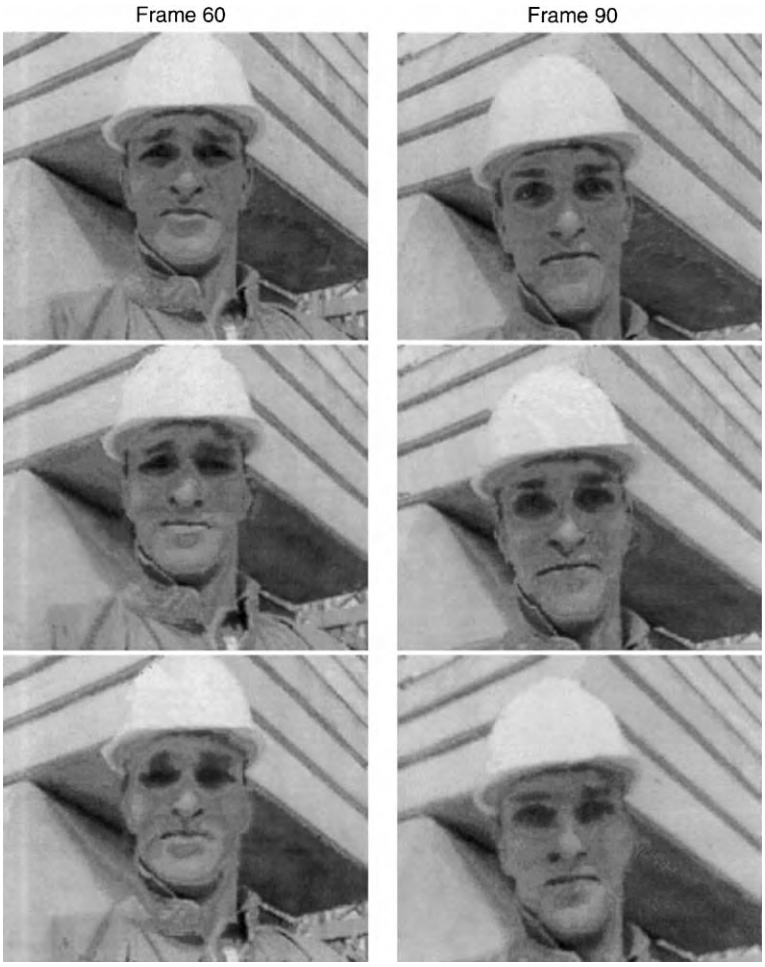

Fig. 14. Coder performance for Foreman sequence at different bit-rates. In the order from top to down, original frames and coded frames at 100 and 65 kbit/s, respectively.

the motion model; for example if the coefficients of the second-order terms are all removed, the model becomes an affine case. The last column of Table 5 implies that the full quadratic model is needed in two-thirds of the time while in one-third of the cases simpler models can be used.

We illustrate how the performance of the coding scheme changes in time by Figs. 19 and 20, where the percentage of MF regions and the PSNR values for Miss America, Claire, Carphone and Foreman sequences, are displayed as a function of time at 23, 28, 79 and 65 kbit/s, respectively. Carphone and Foreman graphs in Fig. 19 reveal that there may be fluctuations in the percentage of MF regions over time.

Simulation results indicate that the subjective quality is better than that of the schemes that rely on spatial segmentation [27,3]. The reason is that in these contour-texture coding schemes, the emphasis is on the problem of object tracking, giving relatively less importance to the resulting quality. The performance remains very dependent on the accuracy of the spatial segmentation, which in turn also necessitates a considerable effort and bit budget for the compensation of the contour prediction errors resulting from the projection.

We have also performed experiments in order to justify the use of quadratic model vis-à-vis affine motion model. The motion estimation scheme, the decision criteria employed during the segmentation phase, and the

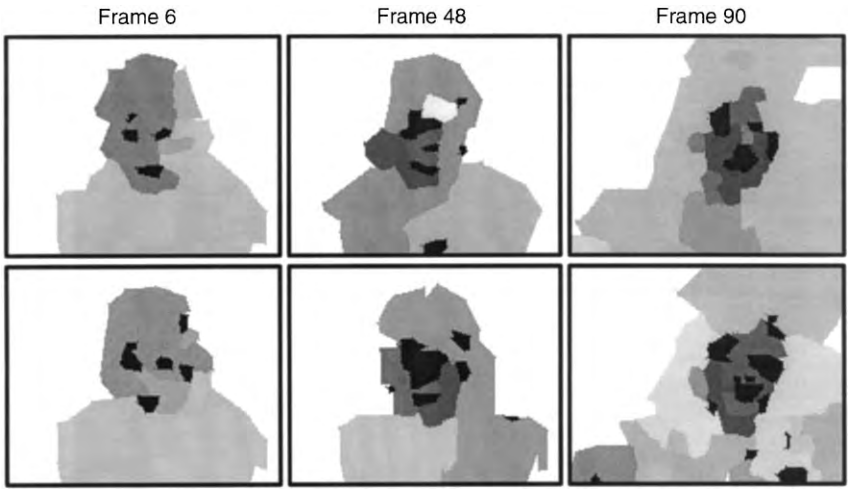

Fig. 15. Polygonized segmentation maps of Miss America sequence at 16 kbit/s (the first row) and 23 kbit/s (the second row).

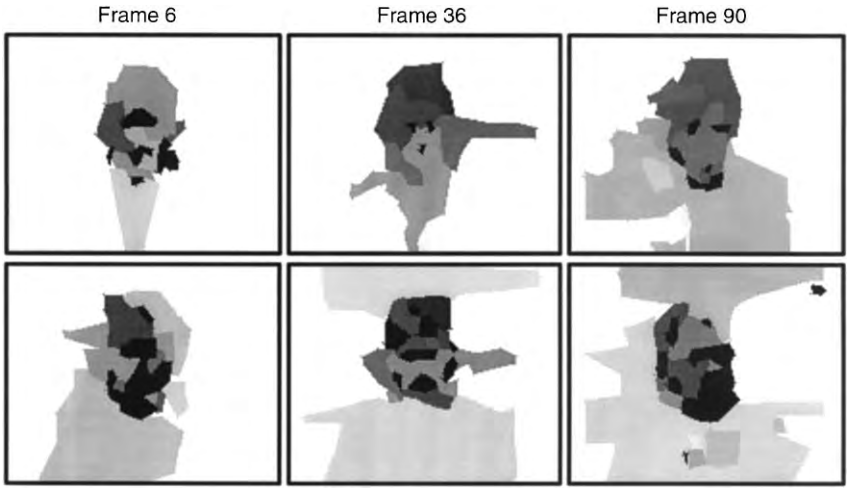

Fig. 16. Polygonized segmentation maps of Claire sequence at 17 kbit/s (the first row) and 28 kbit/s (the second row).

parameter coding strategy have all been kept the same for both cases. The only difference was that the affine model had only six parameters to be estimated and the coefficients of the second-order terms were assumed to be zero. During the experiments, we have observed that affine modeling may sometimes lead to a better motion representation for an object region, depending upon the size, the true motion, and the surface geometry of the region. This is actually due to the fact that the dimension of the optimization problem in affine case is smaller than that of the quadratic model, which in turn reduces the complexity and increases the robustness of the optimization. Thus, in order to make use of affine modeling in the cases where it is more appropriate for motion modeling compared to quadratic transform, we have employed a joint strategy so that during the motion estimation-segmentation scheme, both affine and quadratic modeling have been considered and accomplished separately for each motion estimation, and the one which results in a smaller mean squared error has been chosen as the model of that coding instant. Recall that in the previous

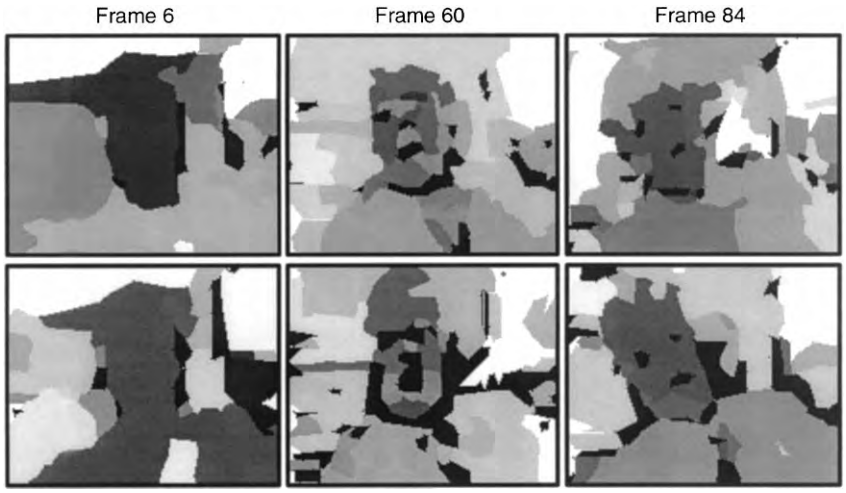

Fig. 17. Polygonized segmentation maps of Carphone sequence at 50 kbit/s (the first row) and 79 kbit/s (the second row).

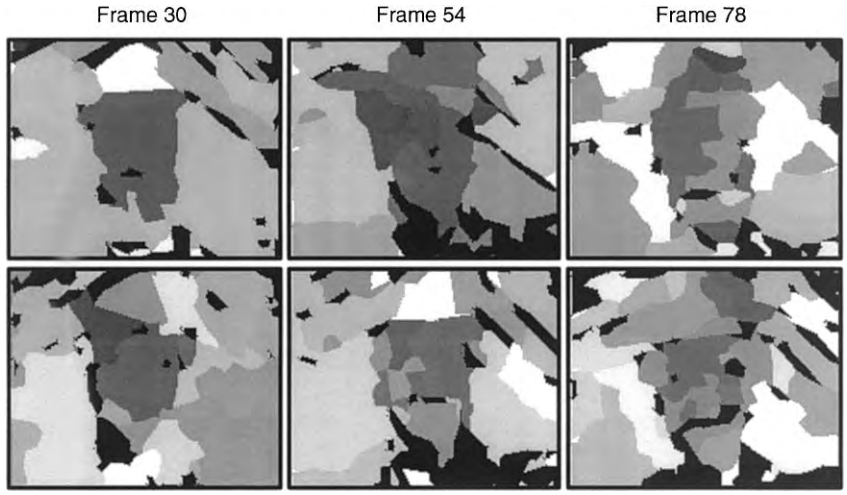

Fig. 18. Polygonized segmentation maps of Foreman sequence at 65 kbit/s (the first row) and 100 kbit/s (the second row).

experiments of which the results have already been presented in Tables 1–4, we had used purely the quadratic model. The performance figures of the coding scheme with the joint motion estimation strategy vis-à-vis those with purely affine model are compared in Table 6. The model failure regions in the initial partition of the image window from where the region growing process takes start, has been forced to be of equal size in two cases so that the comparison is not influenced by initial conditions. In our experiments, the size of model failure regions was initially 10% of the whole image window for Miss America and Claire sequences, and 20% for Carphone and Foreman sequences. The percentage of the number of MC regions which finally ended up with a purely affine model in the case of joint affine-quadratic modeling varied between 10% and 15% for different sequences. The corresponding bit expenditures for shape, motion and color are given in Table 7.

In Table 6, we observe that the transmission bit-rates are higher than those of the previous experiments, and in turn the coding quality is better. The reason of this is that the shape information in this new set of

Table 1
Quantitative performance figures of the coding scheme for Miss America sequence at 5 fr/s (average values per frame). The bit-rate can be found by the sum of shape, motion and color bit expenditures multiplied by the frame rate

Bit-rate (kbit/s)	Shape (bit)	Motion (bit)	Color (bit)	MC (number)	MF (% size)	PSNR (dB)
16	1456	594	1070	9	1.4	34.7
23	1616	560	2292	8	2.7	35.6
37	1893	736	4637	10	6.1	36.2

Table 2
Quantitative performance figures of the coding scheme for Claire sequence

Bit-rate (kbit/s)	Shape (bit)	Motion (bit)	Color (bit)	MC (number)	MF (% size)	PSNR (dB)
17	1281	628	1397	9	1.5	34.8
28	1774	876	2800	12	2.6	36.3
36	1773	887	4486	12	4.3	37.3

Table 3
Quantitative performance figures of the coding scheme for Carphone sequence

Bit-rate (kbit/s)	Shape (bit)	Motion (bit)	Color (bit)	MC (number)	MF (% size)	PSNR (dB)
50	3856	1866	4100	25	4.9	29.7
79	4165	1894	9375	25	10.3	31.5

Table 4
Quantitative performance figures of the coding scheme for Foreman sequence

Bit-rate (kbit/s)	Shape (bit)	Motion (bit)	Color (bit)	MC (number)	MF (% size)	PSNR (dB)
65	3895	1899	6955	24	8.6	30
100	4426	2037	13179	26	14.6	31.2

Table 5
Efficiency of the motion parameter coding scheme: entropy and resulting average bit-lengths for the 12 quadratic motion parameters, and percentage of the removed coefficients

Sequence	Entropy	Average bit length	Removed coefficients
Miss America	5.25	5.81	37%
Claire	5.48	5.97	37%
Carphone	5.76	6.21	33%
Foreman	6.00	6.66	29%

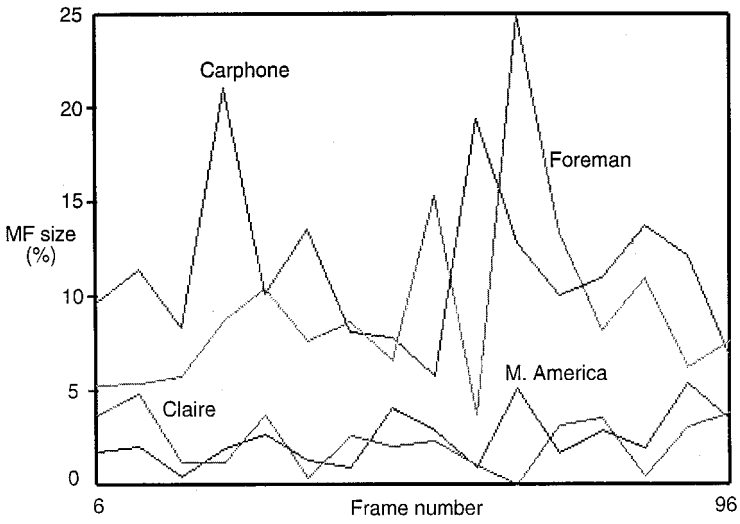

Fig. 19. MF region sizes in individual frames, as a percentage of the QCIF size.

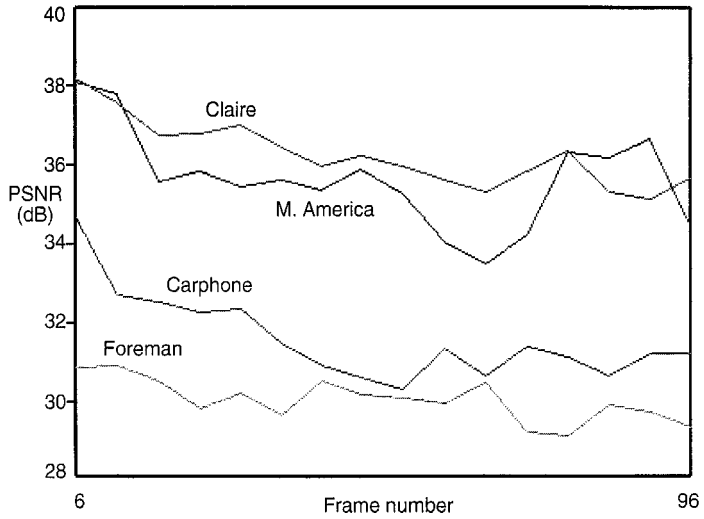

Fig. 20. PSNR values for individual frames.

experiments has been coded with a higher precision in order to have a more accurate comparison which is as free as possible of errors that may result from shape approximation and elimination of very small regions. The PSNR figures achieve an improvement of 0.7 dB in performance for Miss America and Foreman sequences, and 0.4 dB for Claire and Carphone sequences. We should remind that the PSNR value for each image is calculated over the whole frame. Thus although the PSNR improvements seem small, they correspond to regions or objects of the scene for which the motion can be modeled better by a second-order model. The cost of this improvement in PSNR figures is the slight increase in the resulting bit-rates which originates from the additional coding load of the second-order motion model parameters as observed in Table 7. However, this additional cost is tolerable for the resulting PSNR improvements.

Table 6
Comparison of the performance figures of the coding scheme for purely affine and joint affine-quadratic motion estimation

Sequence	Purely affine				Joint affine-quadratic			
	Bit-rate (kbit/s)	MC (number)	MF (% size)	PSNR (dB)	Bit-rate (kbit/s)	MC (number)	MF (% size)	PSNR (dB)
Miss America	37.9	16.4	4.6	36.8	38.6	17.2	4.2	36.9
Claire	35.5	17.1	3.1	37.3	38.0	17.6	3.0	37.5
Carphone	78.9	38.5	7.2	32.9	80.6	41.9	6.3	33.4
Foreman	108.5	48.7	14.3	32.1	110.2	50.5	12.1	32.2

Table 7
Bit expenditures per frame for purely affine and joint affine-quadratic motion estimation

Sequence	Purely affine				Joint affine-quadratic			
	Bit-rate (kbit/s)	Motion (bits)	Shape (bits)	Color (bits)	Bit-rate (kbit/s)	Motion (bits)	Shape (bits)	Color (bits)
Miss America	37.9	558	3257	3775	38.6	954	3320	3452
Claire	35.5	622	3082	3404	38.0	1339	3116	3149
Carphone	78.9	1369	8457	5963	80.6	2608	8223	5294
Foreman	108.5	1965	9485	9995	110.2	3440	9604	8600

6. Conclusions

A novel video coding scheme, in the VLBR range, has been advanced, that exploits the temporal redundancy by a combination of quadratic motion modeling, region growing segmentation and an efficient polygonization scheme. The result is an object-oriented coding scheme, whose advantages and novelties can be summarized as follows:

1. The segmentation problem is formulated as an optimization problem aiming to find the optimal description of the scene in the rate-distortion sense in terms of model compliance and model failure objects defined with their motion, shape and color parameters. This defines a joint problem of motion estimation and segmentation, which has been solved near optimally via an iterative procedure.
2. The object hierarchy [14,4] has been inverted in the sense that the segmentation proceeds from smaller objects to larger ones. By making use of the motion parameters estimated interactively with the segmentation process, a region growing process is started which iteratively redefines the object boundaries and enlarges the initial object seed blocks. The reversed hierarchy and region growing process result in a more flexible segmentation scheme as compared to those in [14,4], by being able to identify more generic motion segments. Thus it is applicable to image sequences of any content, i.e., not necessarily limited to head-and-shoulder scenes.
3. The object motion is represented by the quadratic transform model [16,4], which is a more general and therefore flexible polynomial model as compared to the other lower order polynomial representations, namely the translation and the affine transformation. Quadratic transform provides a representation for the rigid motion and linear deformation of objects with quadratic facets, thus avails us of the possibility to obtain an accurate motion modeling for larger and more complex objects, which is favorable for both coding cost and quality considerations.

4. The performance of quasi-Newton estimation of the quadratic transform coefficients is improved by a combined multiresolution and multiframe analysis. The motion information in the skipped frames is incorporated so that the motion parameters evolve temporally. The multiframe analysis tends to increase the coherency of the objects resulting from the motion segmentation with the true objects, hence better object tracking can be realized.
5. Efficient coding of the object shape is crucial since object-oriented methods have to allocate a considerable part of the available bit budget for shape representation unlike the block-based coders such as H.263. In this work, a context-base shape coding technique is proposed. The context-based denotes here the fact that the shape coding takes into account not only the geometric shape distortion, but also the image synthesis error. In other words, during the shape coding process, the segmentation map is reconsidered and refined so that spurious details of the map are omitted to achieve a more efficient coding.

In conclusion, a complete object-oriented codec has been proposed. Further issues to be addressed in future research list as follows:

The quality performance of our codec is quite good at very low bit-rates when applied to image sequences with limited motion content as in Claire and Miss America sequences. However, the size of model failure regions and the number of model compliance regions increase, as in the case of Foreman and Carphone sequences, the performance deteriorates disturbingly at the very low bit-rates, or alternatively, for satisfactory quality higher speeds are needed. Simulation results show that a considerable part of the bit budget must be allocated to the shape information in this case.

The shape coding efficiency can be further improved by executing the shape coding algorithm simultaneously for each object region in the scene. This puzzle tree organization of regions can be more advantageous in reducing shape redundancy by coding the common boundaries only once.

A second avenue of improvement could be the use of so called Voronoi or Dirichlet diagrams [1,2,20], which can efficiently be utilized in the object tracking problem. A Voronoi diagram is a construct of image points each of which defines a convex polygonal territory which is the region of the image plane nearer to it than to any other data point. Such a representation corresponds to a diagram of neighboring regions which can easily be modified by adding or deleting points. Thus, in the context of object-oriented video coding, it can be utilized for the representation of a segmentation map at different resolution levels and for its modification so as to adapt to the temporal changes in the scene. Once a Voronoi diagram is constructed for the initial frame, the defining points (the seeds) can be projected to the consecutive frames via estimated motion parameters so that the projected points define a new segmentation map, which then can be modified by adding or deleting points mainly at the object boundaries. The mapping of a Voronoi diagram is quite robust, which does not yield ambiguous or conflicting pixels, or nonconvex overlapping regions. Thus our pixelwise region growing technique can be extended so as to proceed more smoothly in terms of Voronoi polygons, resulting hopefully in regions with less spurious details.

References

[1] E. Bertin, H. Bischof, P. Bertolino, Voronoi pyramids controlled by Hopfield neural networks, Comput. Vision Image Understanding 63 (3) (1996) 462–475.
[2] A. Bowyer, Computing Dirichlet tessellations, Comput. J 24 (2) (1981) 162–166.
[3] P. Brigger, Morphological shape representation using the skeleton decomposition: application to image coding, Ph.D. Dissertation, Ecole Polytech. Federale de Lausanne, These No 1448, 1995.
[4] N. Diehl, Object-oriented motion estimation and segmentation in image sequences, Signal Processing: Image Communication 3 (1991) 23–56.
[5] J.-L. Dugelay, H. Sanson, Differential methods for the identification of 2D and 3D motion models in image sequences, Signal Processing: Image Communication 7 (1995) 105–127.
[6] M. Eden, M. Kocher, On the performance of a contour coding algorithm in the context of image coding. Part I: Contour segment coding, Signal Processing 8 (1985) 381–386.

[7] R. Fletcher, Practical Methods of Optimization, Wiley, New York, 1987.
[8] H. Freeman, On the encoding of arbitrary geometric configurations, IRE Trans. Electron. Comput. EC-10 (1) (1961) 260–268.
[9] M. Gilge, T. Engelhardt, R. Mehlan, Coding of arbitrarily shaped image segments based on a generalized orthogonal transform, Signal Processing: Image Communication 1 (1989) 153–180.
[10] C. Gu, M. Kunt, Very low bit-rate coding using multi-criterion segmentation, in: First IEEE International Conference on Image Processing, Vol. 2, 1994, pp. 418–422.
[11] R.M. Haralick, S.R. Sternberg, X. Zhuang, Image analysis using mathematical morphology, IEEE Trans. Pattern Anal. Mach. Intell. 9 (4) (1987) 532–549.
[12] M. Hötter, Object-oriented analysis–synthesis coder based on moving two-dimensional objects, Signal Processing: Image Communication 2 (4) (1990) 409–428.
[13] M. Hötter, Predictive contour coding for an object-oriented analysis-synthesis coder, in: IEEE International Symposium on Information Theory, 1990, p. 75.
[14] M. Hötter, R. Thoma, Image segmentation based on object oriented mapping parameter estimation, Signal Processing 15 (3) (1988) 315–334.
[15] S. Jeannin, On the combination of a polynomial motion estimation with a hierarchical segmentation based video coding scheme, in: IEEE International Conference on Image Processing, Lausanne, Vol. 2, 1996, pp. 489–492.
[16] M. Karczewicz, J. Nieweglowski, P. Haavisto, Motion estimation and representation for arbitrarily shaped image regions, in: IEEE International Conference on Image Processing, Vol. 2, 1995, pp. 197–200.
[17] M. Kunt, A. Ikonomopoulos, M. Kocher, Second generation image coding techniques, Proc. IEEE 73 (1985) 549–575.
[18] F. Marques, P. Brigger, A. Gasull, C. Gu, F. Meyer, C. Oddou, Contour coding, Tech. Rep., Universitat Polytecnica de Barcelona (UPC), 1993.
[19] F. Marques, P. Salembier, M. Pardas, R. Morros, I. Corset, S. Jeannin, B. Marcotegui, F. Meyer, A segmentation-based coding system allowing manipulation of objects (SESAME), in: IEEE International Conference on Image Processing, Lausanne, Vol. 2, 1996, pp. 121–124.
[20] A. Montanvert, P. Meer, A. Rosenfeld, Hierarchical image analysis using irregular tessellations, IEEE Trans. Pattern Anal. Mach. Intell. 13 (4) (1991) 307–316.
[21] H.G. Musmann, M. Hötter, J. Ostermann, Object-oriented analysis–synthesis coding of moving images, Signal Processing: Image Communication 1 (2) (1989) 117–138.
[22] Y. Nakaya, H. Harashima, Motion compensation based on spatial transformations, IEEE Trans. Circuits Systems Video Technol. 4 (3) (1994) 339–356.
[23] J. Ostermann, Object-based analysis-synthesis coding (OBASC) based on the source model of moving flexible 3-D objects, IEEE Trans. Image Processing 3 (5) (1994) 705–711.
[24] F. Pereira, R. Koenen, Requirements and applications for MPEG 4, Tech. Rep. 711, ISO/IEC JTC1/SC29/WG 11, 1994.
[25] W.H. Press, B.P. Flannery, S.A. Teulosky, W.T. Vetterling, Numerical Recipes in C, Cambridge University Press, Cambridge, 1988.
[26] P. Salembier, M. Pardas, Hierarchical morphological segmentation for image sequence coding, IEEE Trans. Image Processing 3 (5) (1994) 639–651.
[27] P. Salembier, L. Torres, F. Meyer, C. Gu, Region-based video coding using mathematical morphology, Proc. IEEE 83 (6) (1995) 843–857.
[28] H. Sanson, Joint estimation and segmentation of motion for video coding at low bit-rates, in: COST 211ter European Workshop on New Techniques for Coding of Video Signals at very Low Bitrates, 1993, Paper 4.4.
[29] H. Schiller, M. Hötter, Investigations on colour coding in an object-oriented analysis–synthesis coder, Signal Processing: Image Communication 5 (1993) 319–326.
[30] G.M. Schuster, A.K. Katsaggelos, An optimal lossy segmentation encoding scheme, in: Proceedings of the Conference on Visual Communication and Image Processing, Vol. 2727, 1996, pp. 1050–1061.
[31] G.M. Schuster, A.K. Katsaggelos, An efficient boundary encoding scheme which is optimal in the rate distortion sense, in: IEEE International Conference on Image Processing, Lausanne, Vol. 2, 1996, pp. 77–80.
[32] T. Sikora, Low complexity shape-adaptive DCT for coding of arbitrarily shaped image segments, Signal Processing: Image Communication 7 (1995) 381–395.
[33] P. Willemin, T. Reed, M. Kunt, Image sequence coding by split and merge, IEEE Trans. Communication 39 (12) (1991) 1845–1855.
[34] Y. Yemez, Object-oriented video coding based on region growing motion segmentation, Ph.D. Dissertation, Electrical and Electronics Eng. Dep., Boğaziçi University, 1997.
[35] Y. Yemez, B. Sankur, E. Anarım, Region growing motion segmentation and estimation in object-oriented video coding, in: IEEE International Conference on Image Processing, Lausanne, Vol. 2, 1996, pp. 521–524.
[36] Y. Yemez, B. Sankur, E. Anarım, An object-oriented video codec based on region growing motion segmentation, in: IEEE International Conference on Image Processing, 1997, pp. 444–447.