A Region-based Video Coder Using Edge Flow Segmentation and Hierarchical Affine Region Matching

Debargha Mukherjee, Yining Deng, and Sanjit K. Mitra

Department of Electrical and Computer Engineering,
University of California, Santa Barbara, CA 93106.
Email: {debu,deng,mitra}@iplab.ece.ucsb.edu

ABSTRACT

The essential motivations, towards an object-based approach to video coding, include possible object-based interactivity and functionality, and the elimination of blocking or mosquito artifacts that typically occur in a block-based coding scheme. In this work we present a region-based video coder which uses a segmentation map obtained from the previous reconstructed frame, thereby eliminating the need to transmit expensive shape information to the decoder. While the inspiration for this work is derived from previous work by Yokoyama et al[1,2], there are major differences between our work and the earlier effort, in the segmentation scheme employed, the motion model, and the handling of overlapped and uncovered regions. We use an edge flow based segmentation scheme[3], which appears to produce consistent segmentation results over a variety of natural images. Since it combines luminance, chrominance and texture information for image segmentation, it is well suited to segment real world images. For motion compensation, we choose an affine model, and use hierarchical region-matching for accurate affine parameter estimation. Heuristic techniques are used to eliminate overlapped and uncovered regions after motion compensation. Extensive coding results of our implementation are presented.

Keywords: object-based video coding, edge flow, spatial segmentation, spatio-temporal segmentation, hierarchical region matching, affine motion model, overlapped regions, uncovered regions, residual error coding.

1. INTRODUCTION

The progressive convergence of telecommunications, computer, and TV/film industries, in the recent years has resulted in widespread research activity in the area of *object-based* or *region-based* video coding[5,6,7,8,9]. Unlike traditional block-based coding algorithms that operate on units of regular square blocks called *macroblocks*, region-based coders operate on units of arbitrary shaped regions that represent semantically consistent objects. The motivation in adopting such an approach is two-fold. First, the use of arbitrarily shaped segments based on similarity in luminance, chrominance, texture, and motion can eliminate blocking or mosquito artifacts that frequently occur in block-based coders, by making the coding discontinuities coincide with the natural boundaries in images. Although the encoder now needs to code and transmit additional segmentation information to the decoder, apart from the motion and motion compensated residual information, it is hoped that the gain in coding efficiency, in particular the subjective performance, will outweigh the decrease in efficiency due to the additional bits transmitted. Second and more important in today's context, the use of an object-based methodology in coding, opens up possibilities in object based interactivity and functionality. It was this compelling drive that made the Motion Pictures Experts Group (MPEG) initiate the MPEG-4 standardization phase in 1994, with the release of the standard targeted for late 1998.

The MPEG-4 Video Standard Verification Model[10,11] envisages a layered representation for a video sequence. A scene is composed by several video object layers overlaid on one another. The attempt is to encode a video sequence in a way that allows separate decoding and reconstruction of the objects and manipulation of the original scene by simple operations on the bit stream. The bit stream will be object layered, and the shape and transparency of each object, as well as its spatial coordinates and parameters describing object scaling, rotation, and other related attributes, are described in the bit stream of each object layer. It is important, however, to realize that the MPEG-4 Video Verification Model is not an independent object-based coder in the strict sense. It only provides a framework to code and transmit efficiently the information in each object layer, when the segmentation information is already available in a 3D mask known as the *alpha plane*. It does not actually address the issue of segmenting an image sequence into layers. Since automatic segmentation of image sequences is still an unsolved problem, the use of the Verification Model to encode and manipulate real world scenes is limited in scope.

True object based coders generate segmentations for each frame of a sequence automatically, and transmit the shape information for each region to the decoder along with information on its motion, and motion compensated residual error. Even for the best of object-based coders, the fraction of the bitrate spent in transmitting shape information to the decoder is about 20-30%. An alternative approach to object-based coding is to have the decoder derive the segmentation of the scene from the previous reconstructed frame[1,2], thereby eliminating the need to transmit expensive shape information. In this work, we develop algorithms to assemble a coder of the latter category. One of the probable reasons why there has been little research on this class of coders is the drawback of excessive decoder complexity, owing to the fact that the reconstructed frame needs to be segmented in order to decode the next frame. However, with the current pace of advancement in the semiconductor industry, this concern will soon be insignificant.

The rest of this paper is organized as follows. Section 2 presents a schematic of the coder. We use an edge flow based scheme[3,4] that combines color and texture information for spatial segmentation. Section 3 briefly discusses the segmentation scheme. For accurate and robust affine parameter estimation, a hierarchical region matching algorithm, that is better suited to estimate large motion than traditional gradient-based schemes, is used. We describe the scheme in Section 4. In Section 5, the step-by-step heuristics used to handle the overlapped and uncovered regions, are discussed. Section 6 presents a brief note on the residual error coding schemes used after motion compensation and cleaning. Section 7 discusses the issue of segmentation propagation from one frame to the next. In Section 8 we present the coding results obtained with our coder on QCIF video sequences. Section 9 concludes the paper with a note on future directions.

2. SCHEMATIC OF THE CODER

Figure 1 presents a schematic of the proposed encoder. In the figure, I_k represents an input frame to be encoded. The previous reconstructed frame $R_{k-1}$is already available to the encoder. The Segmentation module performs a spatial segmentation of R_{k-1}, and generates a *spatial segmentation map*. The Motion Estimation and Region Merging module estimates forward affine parameters for all the regions the frame is segmented into, and merges adjacent segments if their motion parameters are sufficiently close. The motion parameters are then re-estimated for the merged segments. The estimated motion parameters along with the merging information is transmitted to the decoder. The motion-merged segmentation map is referred to as the *spatio-temporal segmentation map*. Note that the motion parameters are estimated using the raw current and previous frames, because we do not want our motion estimation to be affected by the distortions in the previous reconstructed frame. The Motion Compensation module uses the spatio-temporal segmentation map of the previous reconstructed frame, and the motion parameters associated with each region to produce a crude prediction Q_k of the current frame. Q_k is not a usable prediction because it has a number of regions where we have more than one prediction, called *overlapped regions*, and a number of regions where we have no predictions at all, called *uncovered regions*. The Overlapped and Uncovered Region Cleaning module performs heuristic operations to clean up these regions so as to obtain a valid prediction for each pixel of the current frame. The predicted frame P_k obtained after these operations constitute the final predicted frame. The rest of the operations are standard. The current frame is subtracted from the predicted frame to obtain an error frame E_k, which is then transform coded to $\hat{E}_k$. The predicted frame is then added to it to obtain the new reconstructed frame R_k.

In the next few sections we will visit each of the modules in detail. There is one issue, however, that the block diagram in Figure 1 does not address. It is possible to reduce the computational complexity at the cost of loss in coding efficiency by performing the segmentation only occasionally, rather than once for every frame coded. It will then be necessary to propagate the segmentation information from one frame to the next as accurately and economically as possible. The problem is by no means simple, especially because of the heuristics used to clean up the uncovered and overlapped regions in the motion compensated frame. Moreover, the propagation problem is linked with the choice of the residual error coding scheme used. We will address the issues pertaining to this problem in detail in Section 7.

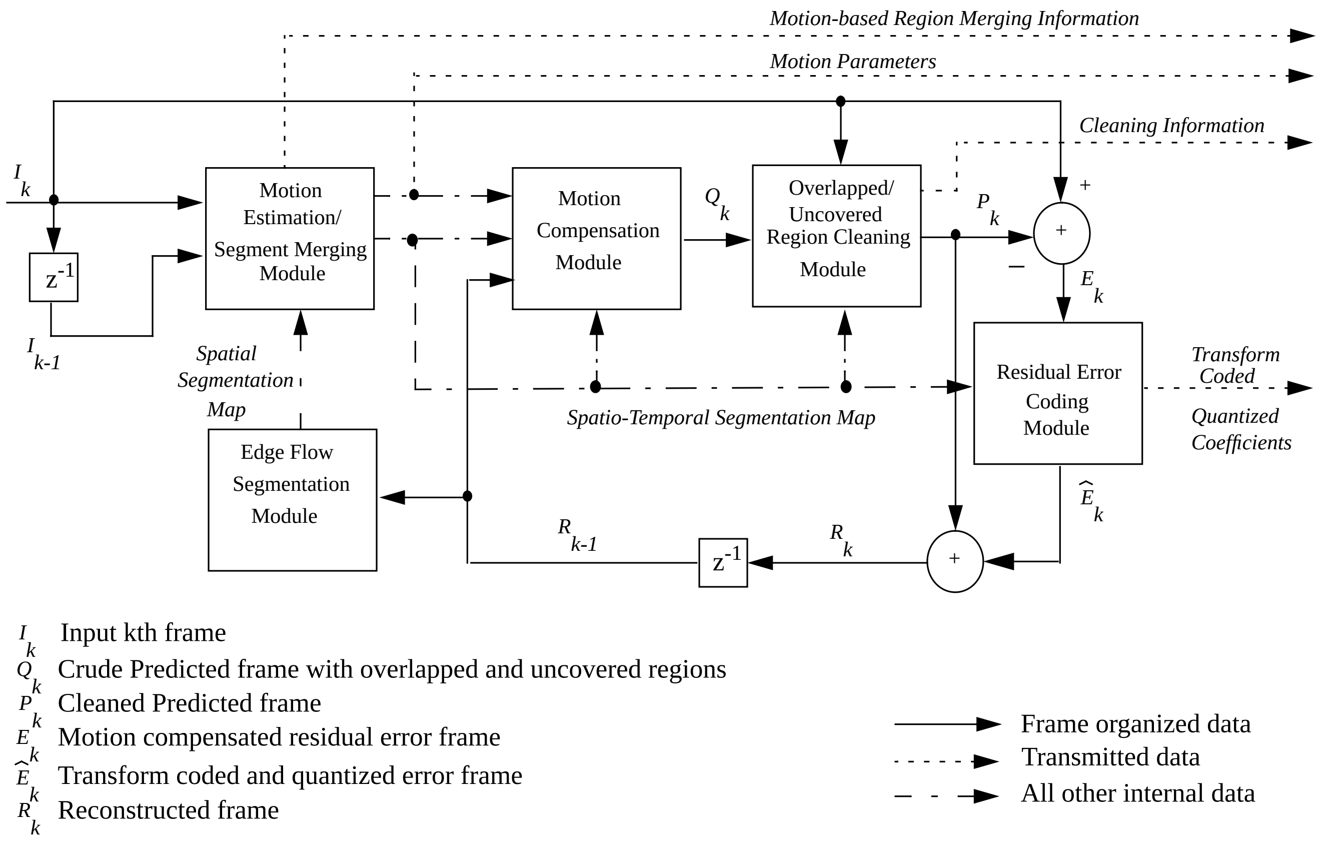

Motion-based Region Merging Information
Motion Parameters
Cleaning Information
I_k
z^{-1}
I_{k-1}
Motion Estimation/ Segment Merging Module
Motion Compensation Module
Q_k
Overlapped/ Uncovered Region Cleaning Module
P_k
+
+
−
E_k
Residual Error Coding Module
Transform Coded
Quantized Coefficients
Spatial Segmentation Map
Spatio-Temporal Segmentation Map
Edge Flow Segmentation Module
$\hat{E}_k$
R_{k-1}
z^{-1}
R_k
+
I_k Input kth frame
Q_k Crude Predicted frame with overlapped and uncovered regions
P_k Cleaned Predicted frame
E_k Motion compensated residual error frame
$\hat{E}_k$ Transform coded and quantized error frame
R_k Reconstructed frame
Frame organized data
Transmitted data
All other internal data

Figure 1. Block-diagram of the proposed coder.

3. EDGE FLOW BASED SEGMENTATION

An important aspect of the proposed codec is the edge flow based segmentation algorithm[3] that combines luminance, chrominance, and texture information to detect boundaries between regions. The decoder duplicates the segmentation procedure for decoding. The segmentation is performed on the previous reconstructed frame that is available both to the encoder and the decoder. Since the reconstructed frame is very noisy, it is necessary to smooth the image with a Gaussian window before segmentation. Also, for QCIF images, the subsampled chrominance components are interpolated to full size, and converted to RGB components before presenting to the segmentation algorithm. A brief outline of the segmentation scheme is presented below.

Since natural images contain discontinuities in color, texture or both, a good segmentation algorithm should consider these different attributes together in computing a partition of the image. Ma and Manjunath[3] presented a general framework for boundary detection based on a predictive coding model. The basic idea is to identify and integrate the direction of change in color, texture, and filtered phase discontinuities at each image location. From this, an edge flow vector which points to the closest boundary is constructed. The edge flow is iteratively propagated to its neighbor if the edge flow of the corresponding neighbor points in a similar direction. The edge flow stops propagating if the corresponding neighbor has an opposite direction of edge flow. In such a case, the two image locations have their edge flows pointing at each other indicating the presence of a boundary between the pixels. Once the flow propagation reaches a stable state, all the local edge energies will be accumulated at the nearest image boundaries. The boundary energy is then defined as the sum of the flow energies from either side of the boundary. After boundary detection, disjoint boundaries are connected to form closed contours, thus partitioning the image into a number of regions. This is followed by a region merging algorithm. Region merging utilizes similarity in color, texture of neighboring regions, as well as the length of the boundary between those regions to decide whether to merge regions or not. Regions smaller than a certain minimum size threshold are forcefully merged with their most similar neighbor. The segmentation module ouputs a segmentation map for use with the motion estimation module. If there are N regions in all, the spatial segmentation map can be regarded as a collection of sets $S_{k-1} = \{S_{k-1,i}, i = 1, 2, \ldots, N\}$, where each set $S_{k-1,i}$ is a collection of pixels (x,y) comprising the ith region in the (k-1)th frame.

4. HIERARCHICAL AFFINE REGION MATCHING FOR MOTION COMPENSATION

The success of any video coding algorithm, block-based or object-based, depends, to a large extent, on the efficiency of the motion compensation scheme used for exploitation of the temporal redundancies in a video sequence. While block-based coders typically use a simple displacement model for block motion, object-based video encoders typically use 6-parameter (affine), 8-parameter (perspective), and sometimes even more complicated motion models. However, the advantage of an accurate motion representation by the use of a complicated motion model is often far outweighed by the difficulty in estimating the parameters. Gradient based or optical-flow based methods are typically used to estimate the parameters. Such methods run the risk of converging at a local minima, are very sensitive to noise, and in general fail if the motion is large. Furthermore, for most sequences, the advantage of using a complicated model is small, and the additional bits spent in transmitting more parameters may not be justified even in a rate-distortion sense. We adopt a very simple affine motion model with only zoom, rotation and translation parameters, and use a hierarchical region matching algorithm for parameter estimation. Hierarchical region matching is analogous to hierarchical block-matching[13] which has been used with considerable success over the years by many researchers. We formalize the scheme below.

Under the affine motion model, it is assumed that the regions obtained by segmentation of the previous reconstructed frame can be affine transformed to obtain the luminance and chrominance components of the pixels in the current frame with reasonable accuracy. Given a region map S_{k-1} of the previous (k-1)th frame derived by segmentation of the previous reconstructed frame, our objective is to find, for each region $S_{k-1,i}$, a linear transformation matrix A_i, and a displacement vector d_i, such that the luminance and chrominance values of the region when projected on the current frame, denoted by $\hat{S}_{k,i}$, match those of $S_{k-1,i}$, as closely as possible. If (m,n) is an integer pixel in the ith region $S_{k-1,i}$ of the previous frame, the corresponding non-integer pixel position (m', n') in the corresponding region $\hat{S}_{k,i}$ in the current frame, is obtained as follows:

$$\begin{bmatrix} m' \\ n' \end{bmatrix} = A \cdot \begin{bmatrix} m - C_x \\ n - C_y \end{bmatrix} + \begin{bmatrix} C_x \\ C_y \end{bmatrix} + d, \qquad A = \begin{bmatrix} a_1 & a_2 \\ a_3 & a_4 \end{bmatrix}, \qquad d = \begin{bmatrix} d_x \\ d_y \end{bmatrix}, \tag{1}$$

where (C_x, C_y) represents the non-integral centroid of the parent region $S_{k-1,i}$ in the previous frame. The subscript i has been dropped from the variables in (Eq. 1) for simplicity.

We now consider the issue of region matching. The basic idea of the scheme is to try a series of admissible motion parameter sets, and to pick the one that produces the best match between the original region and the motion compensated warped region. Since this involves performing the actual motion compensation for each set of test parameters, and choosing the set that minimizes a distortion measure, it is appropriate to consider the issue of motion compensation first. The problem is tricky because integer pixel locations in the current frame do not correspond to integer locations in the previous frame and vice-versa. Figure 2 illustrates the motion compensation methodology used. The figure shows a parent region $S_{k-1,i}$ in the previous frame, warped by a parameter set A and d, into a region $\hat{S}_{k,i}$ in the current frame. If we describe the smallest rectangle $PRect_{k\text{-}1,i}$ enclosing the region in the previous frame, the rectangle will be warped by the affine transform into a parallelogram $WPlgm_{k,i}$ in the current frame. The nature of the affine transform ensures that the warped parallelogram encloses the warped region entirely. Since the coordinates of the four corners of the rectangle in the previous frame are known, the corner coordinates for the warped parallelogram in the current frame are readily obtained from (Eq. 1). Next, all the integer pixels (p,q) within the parallelogram in the current frame are scanned, and back projected into non-integer pixel positions (p', q') in the previous frame using the parameters A and d, following:

$$\begin{bmatrix} p' \\ q' \end{bmatrix} = A^{-1} \cdot \begin{bmatrix} p - C_x - d_x \\ q - C_y - d_y \end{bmatrix} + \begin{bmatrix} C_x \\ C_y \end{bmatrix}, \qquad A = \begin{bmatrix} a_1 & a_2 \\ a_3 & a_4 \end{bmatrix}, \qquad d = \begin{bmatrix} d_x \\ d_y \end{bmatrix} \tag{2}$$

Now, if $(round[p'], round[q'])$ is included in the parent region $S_{k-1,i}$ in the previous frame, we consider (p,q) to be included in the warped region $\hat{S}_{k,i}$ in the current frame. The predicted value at (p,q) in the current frame is then obtained by bilinear interpolation from the fractional location (p', q') in the previous reconstructed frame.

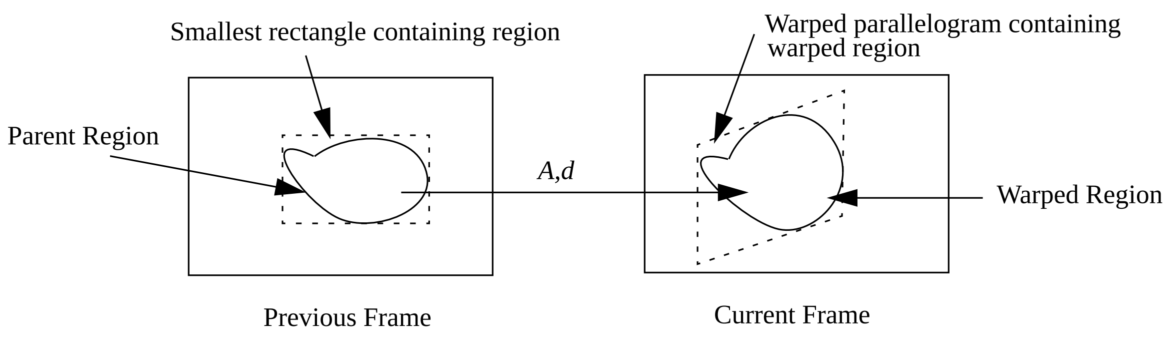

Smallest rectangle containing region
Warped parallelogram containing warped region
Parent Region
A,d
Warped Region
Previous Frame
Current Frame

Figure 2. Illustration of Motion Compensation

Under the motion compensation framework discussed above, the motion estimation problem for the *i*th region $S_{k-1,\,i}$ is formally defined as minimization of the following distortion metric $D_{A,d}$ over all A and d:

$$D_{A,\,d} = \sum \left| I_k(p,q) - I_{k-1}(p',q')_{interpolated} \right|, \quad (p,q) \in WPlgm_{k,\,i} \quad , \ (round[p'], round[q']) \in S_{k-1,\,i} \tag{3}$$

Here the relation between (p, q) and (p', q') are given by (Eq. 2), the I_k and I_{k-1} represent the luminance components of the current and previous input images, and the value at non-integer location (p', q') in the previous frame is obtained by interpolation. Although we use only the luminance component for region matching, it is a simple extension to use the chrominance components as well.

Since the parameter space searched in region matching is fairly large, one concern with the algorithm is excessive search complexity. For example, with the 6-parameter affine model, we need to search a large number of different admissible 6-tuples $(a_1, a_2, a_3, a_4, d_x, d_y)$ in a 6-dimensional parameter space for the best match. However, by choosing the test parameters in a hierarchical fashion, gradually narrowing the search space over a series of increasing resolution images, the complexity of the algorithm can be made more tractable. Both the current and the previous frames are successively lowpass filtered and decimated by a factor of two, in a pyramidal fashion. The segmentation map is correspondingly downsampled in a pyramidal fashion. For each region, we start our search at a level of resolution where the number of pixels included is neither so large that the complexity is excessive, nor so small that the confidence in region matching becomes too low. All admissible parameter sets at the starting resolution level are searched for the best match. Once the search is completed at a certain level of resolution, the best parameter set obtained is used to generate the initial guess for the search in the next higher resolution level. The best displacements in the lower level are doubled, while the parameter set $\{a_1, a_2, a_3, a_4\}$ are used unmodified to obtain the initial guess in the next higher level of resolution. The search is conducted only in a restricted zone around the initial guess in the next higher level. In this manner, a good estimate of the motion parameter set is obtained by only a small search range at full resolution.

To further reduce the search complexity, we used 5-parameter and 4-parameter simplified affine models for our simulations, by restricting the parameter matrix to be a composite of rotation and zoom only. For the 5-parameter affine model, A can be represented as:

$$A = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \cdot \begin{bmatrix} Z_x & 0 \\ 0 & Z_y \end{bmatrix} = \begin{bmatrix} Z_x\cos\theta & Z_y\sin\theta \\ -Z_x\sin\theta & Z_y\cos\theta \end{bmatrix}, \tag{4}$$

while for the 4-parameter model, we further assume $Z_x = Z_y = Z$, so that

$$A = Z \cdot \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}. \tag{5}$$

For both these models, note that the warped parallelogram $WPlgm_{k,i}$ is simplified to a rectangle in the current frame.

Spatially distinct adjacent segments in an image may have similar motion if they belong to the same object. For example, the head and the face of a person, usually has the same motion parameters, even though they may form distinct spatial segments. In such cases, it is reasonable to merge them into one. After the motion parameters have been estimated for each spatially segmented region, we attempt merging adjacent segments if their motion parameters are reasonably close, and they share a reasonably long common boundary. The motion parameters are re-estimated for every pair of such candidates, but the merge is rejected if the increase in distortion with joint motion parameters over separate motion parameters is substantial. Adjacent regions that pass the above criterion are merged, and the merging information is transmitted to the decoder with a few bits. The updated segmentation map, now called the spatio-temporal segmentation map, is denoted by a collection of sets $ST_{k-1} = \{ST_{k-1,i}, i = 1, 2, \dots, M\}$ where each set $ST_{k-1,i}$ is a collection of pixels belonging to region *i*, and $M \le N$, the number of spatially segmented regions. This region map is used for the rest of the coding process. The motion parameters for each of the final regions are also transmitted to the decoder.

5. OVERLAPPED AND UNCOVERED REGIONS

Following the motion compensation method outlined in the previous section, the spatio-temporal segmentation map of the previous frame and the motion parameters are used to obtain a crude prediction Q_k of the current frame. This frame has a number of regions where there exist more than one predictions, and a number of regions where there are no predictions at all. The former are referred to as *overlapped* regions, while the latter are referred to as *uncovered* regions. While the overlapped regions are fairly easy to deal with, the uncovered regions pose a more acute problem. In order to obtain a consistent predicted frame that approximates the input frame closely enough, it is necessary to clean up the overlapped and uncovered regions as best as possible with minimal transmission of overhead information to the decoder. Yokoyama et al[1] used a simple interpolation scheme to obtain a prediction at each pixel location, however, such a scheme is clearly not suitable for large overlapped and uncovered regions. Below we outline a step-by-step procedure for cleaning the overlapped and uncovered regions.

Step 1: All isolated overlapped pixels are cleaned by assigning their luminance and chrominance components to the average of the candidate predictions.

Step 2: This step removes all overlapped regions by ordering all the regions into layers and transmitting the layering information to the decoder. We define an Overlapping Distortion matrix called *OD*, of size $M \times M$, for this purpose. The matrix is initialized to 0, and the elements are filled up as follows. All the overlapped pixels in the crude predicted image are scanned. If there are *K* candidate predictions for a pixel from regions $r_1, r_2, \dots, r_K$ respectively, we compute a measure of the combined distortion in luminance and chrominance for each of them by computing the sum of the absolute difference between the true values and the predicted values. Let the distortions be denoted respectively by $d_1, d_2, \dots, d_K$. Then, for each *i* in $1, 2, \dots, K$, we update the matrix *OD* as follows:

$$OD_{r_i r_j} = OD_{r_i r_j} + d_i, \quad j = 1, 2, \dots, K, \quad j \neq i \tag{6}$$

At the end of the scan, the (*r*,*s*)th element $OD_{r,s}$ of the matrix *OD*, defines in some sense, the total distortion introduced if region *r* is placed over region *s*. Therefore, if $OD_{rs} > OD_{sr}$, region *r* should be designated as a lower layer below region *s*, and vice versa. The absolute difference $|OD_{rs} - OD_{sr}|$ determines the penalty if the layers are reversed in order. Given the filled up *OD* matrix, we transmit the layering order information to the decoder for every pair of regions that overlap, in order of the distortion penalty associated with the pair. Given the layering order for a pair of regions, the decoder can remove the lower layer from contention wherever the two regions contend for predicting the same pixel. By the time the decoder receives all the layering information, all the overlapped regions would be removed. Alternatively, it is possible to stop transmission of the layering information after the reversal penalty goes below a certain threshold. In that case, the prediction for the remaining overlapped regions is obtained by a simple averaging of the candidate predictions. Since the reversal penalty is low, it is reasonable in a rate-distortion sense to save bits by not sending the layering order. At the end of this step, all overlapped regions would be eliminated.

Step 3: All isolated uncovered pixels are cleaned by assigning their luminance and chrominance components to the average of the pixels surrounding them.

Step 4: The rest of the uncovered regions are in general very difficult to handle. We adopt the following procedure to obtain a prediction for all the uncovered regions larger than a certain size threshold. These regions are filled up by backward motion compensation from the previous reconstructed frame by a region matching scheme similar to the one described in

Section 4. The motion parameters for these regions are then transmitted to the decoder. Note that since the region map for the uncovered regions are already available to the decoder, it is not necessary to transmit any shape information.

Step 5: Finally, the remaining small uncovered regions are filled up by extrapolation from the neighboring predictions.

The five steps above generate the final predicted frame P_k in Figure 1.

6. RESIDUAL ERROR CODING

Once the final predicted frame is obtained, the residual error frame is transform coded. We investigated two options for residual error coding. In the first method, DCT-based coding same as that used in H.263[14] is used. Every 16×16 macroblock is divided into six 8×8 blocks (four luminance and 2 chrominance), and discrete-cosine-transformed. The coefficients are quantized by a uniform quantizer with a dead zone around zero, and stepsize proportional to a Quantization Parameter QP in the range 1-31. The quantized coefficients are zigzag scanned, and jointly runlength-entropy coded using the three dimensional Huffman tables in H.263. The second method used a color extension of the wavelet-based Set Partitioning in Hierarchical Trees algorithm proposed by Said and Pearlman. The luminance and the two chrominance components were individually coded by set-partitioning in the wavelet domain. The motivation for using the second scheme was to reduce the blockiness of the reconstructed frame so that it does not affect adversely the spatial segmentation for coding the next frame.

7. SEGMENTATION PROPAGATION

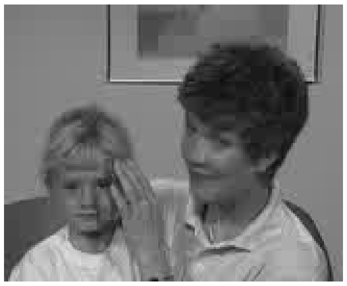
While propagation of the segmentation from frame to frame is not essential to the coding gain in this algorithm, it is necessary for reducing the computational complexity of the encoder and the decoder. In fact, repeating the segmentation for every frame coded is more likely to benefit the rate-distortion performance of the coder than hamper it. However, full spatial segmentation of a frame is expensive, and in order to strike a compromise between efficiency and complexity, it is appropriate to do the spatial segmentation only once in several frames coded. For the rest of the frames, the segmentation is propagated from one frame to the next under certain assumptions. There are two distinct design choices to be considered, and that involves the nature of the residual coding algorithm. In our implementation we have chosen only one, but we include a discussion of the latter for completeness.

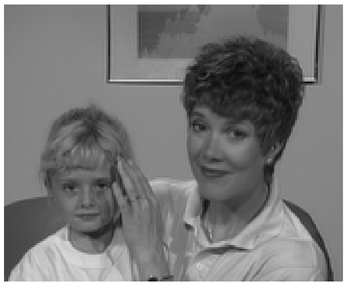
It is to be noted, that when the residual error frame is coded, the segmentation map of the current frame is not known. The first scenario assumes a residual error coding method that does not depend on the segmentation of the current frame, that is non region-based. Both the schemes in Section 6 fall in this category. In such cases, the current frame can be reconstructed before the segmentation map for coding the next frame is needed. Two assumptions are in order before the segmentation from the previous frame can be propagated to the current. First, apart from the uncovered and overlapped pixels, the segmentation map of the crude predicted frame obtained by motion compensation of the previous reconstructed frame, matches closely the true segmentation map. Second, the sequence does not change significantly between two fully segmented frames, so that the number of regions in each frame in between, can be considered to remain the same as that in the last fully segmented frame. Under these assumptions, the problem reduces to reassigning the uncovered and overlapped pixels to one of the regions adjacent to them after residual error coding and current frame reconstruction. The task can be performed fairly easily by region growing on the uncovered and overlapped pixels and assigning the grown regions to one of the surrounding compensated regions based on closeness in luminance and color.

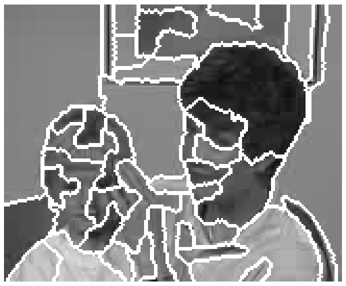
The second scenario utilizes an object-based residual error coding algorithm, which assumes knowledge of the current frame's segmentation map before coding. It is now necessary for the decoder to estimate the current frame's segmentation map without knowing the reconstructed frame. Under the assumptions above, all the uncovered and overlapped pixels in the crude predicted frame must then be assigned to one of the regions. Most overlapped pixels are assigned to regions by the layering information already transmitted. For the large uncovered regions that are filled by backward motion compensation, each pixel is assigned to the region from which it is predicted in the previous frame. If there are overlapped pixels whose candidate predictions have been averaged to obtain a single prediction, they are assigned to the neighboring region to which the average is closest in luminance and color. The isolated uncovered pixels are assigned to the region with the maximum number of pixels surrounding it. Pixels from the uncovered regions that are filled by extrapolation are assigned to one of the surrounding regions to which it is closest in distance. The estimated segmentation map can then be used for object based coding of the residual error. The reconstructed frame obtained after residual error coding is then used to reassign the uncovered and overlapped pixels as in the previous case.

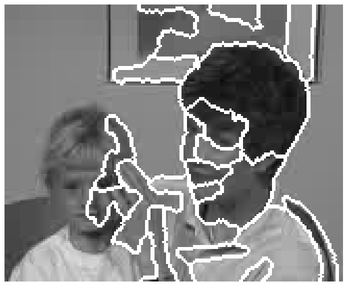
8. RESULTS

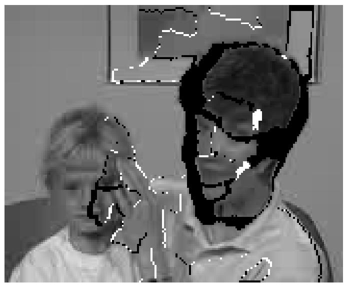
In this section, we present the results of our implementation of the proposed coder on some QCIF video sequences. Figure 3 illustrates the coding algorithm by showing the successive stages involved in coding frame 36 of the Mother_Daughter sequence from the previous reconstructed frame 32, when the sequence is coded at a frame rate of 7.5

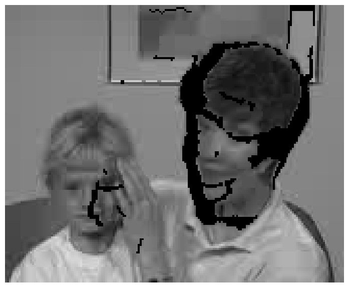

(a) Previous Reconstructed Frame 32

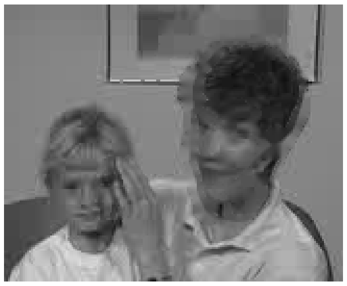

(b) Current Input Frame 36


(c) Edge Flow Spatialy Segmented Frame 32


(d) Spatio-temporally Segmented Frame 32


(e) Crude Predicted Frame 36
(White: Overlapped Regions, Black: Uncovered Regions)


(f) Crude Predicted Frame 36 with Overlapped Regions cleaned
(Black: Uncovered Regions)


(g) Final Predicted Frame 36

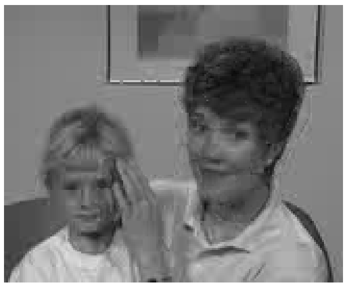
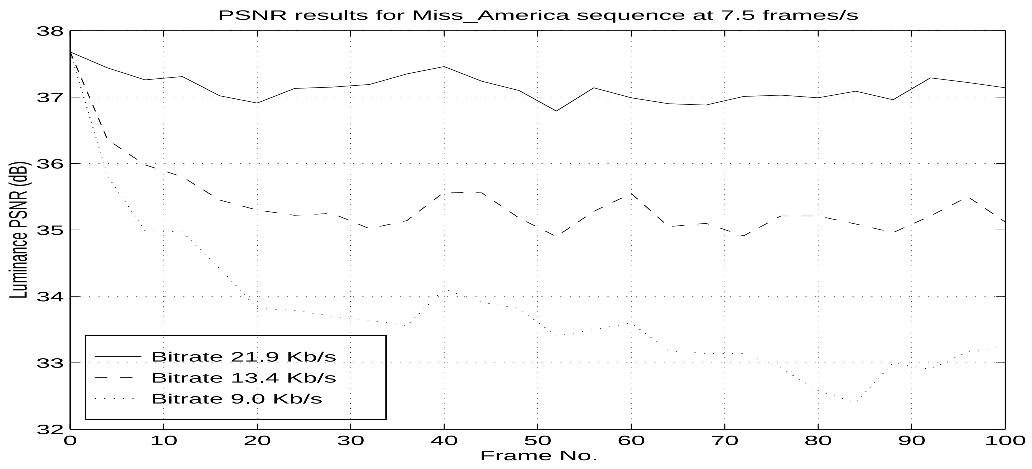

(g) Reconstructed Frame 36 after DCT-based residual coding

Figure 3. Illustration of the coding algorithm. Frame 36 of the mother-daughter sequence is coded from reconstructed Frame 32 at the frame rate of 7.5 frames/s.

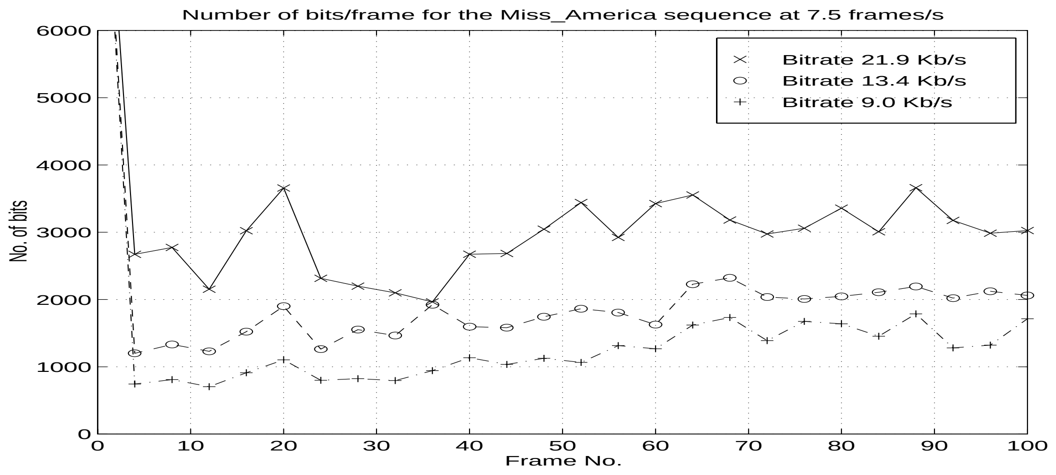
frames/s, and an average bitrate of 24 kb/s. We have purposely chosen these two frames between which the motion is fairly large, to emphasize the various stages in the coding process.

For all the results presented in this section, including the above, motion is estimated at 0.5 pixel accuracy in the range -14 to 14 for displacement, 2 deg accuracy in the range -10 to 10 deg for rotation, and 0.05 accuracy in the range 0.7 to 1.3 for the zoom factor in the 4-parameter affine model in (Eq. 5). A maximum of three levels of decimation are done for hierarchical region matching. DCT-based residual error coding is used. All PSNR results are for the luminance component alone. The PSNR for the chrominance components are between 3-7 dB higher. All the bit rate results exclude the bits spent in INTRA coding the first frame.

Figure 4 shows the evolution of the Luminance PSNR and the bits per frame for the first 100 frames of the Miss_America QCIF sequence coded at 7.5 frames/s (frameskip = 3), for three different bitrates: 21.9 Kb/s, 13.4 Kb/s and 9 Kb/s. Different bit rates are obtained by changing the quantizer parameter in the DCT based residual error coding. The first frame of the sequence in each case is DCT-based INTRA coded using the coding and quantization scheme of H.263. Figure 5 presents the original and the reconstructed frame 92 of the Miss_America sequence when coded at the above bit rates. This frame was chosen because the sequence shows a lot of motion around this frame.



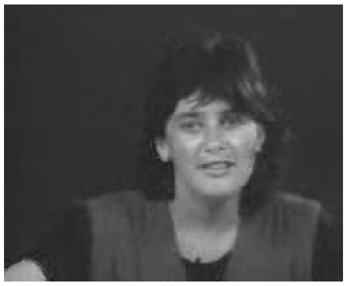
(a) Evolution of Luminance PSNR with Frame Number

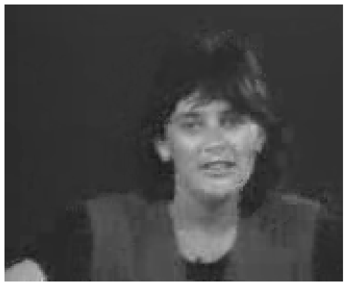


(b) Evolution of Bits per Frame with Frame Number

Figure 4. Evolution of Luminance SNR and Bits/Frame for the first 100 frames of the Miss_America sequence coded at 7.5 frames/s for three bit rates: 21.9 Kb/s, 13.4 Kb/s and 9.0 Kb/s.

(a) Original Frame 92


(b) Reconstructed Frame 92 (21.9 Kb/s)


(c) Reconstructed Frame 92 (13.4 Kb/s)

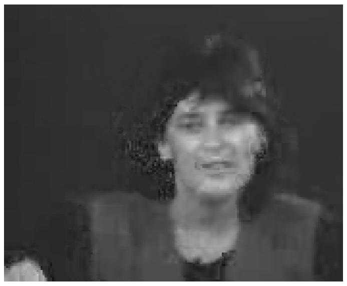

(d) Reconstructed Frame 92 (9.0 Kb/s)

Figure 5. Original and Reconstructed Frame 92 of the Miss_America sequence coded at 7.5 frames/s, for three different bit rates: 21.9 Kb/s, 13.4 Kb/s and 9.0 Kb/s.

Finally we present some results for the Foreman QCIF sequence where the motion is relatively large. Figure 6 shows the average Luminance PSNR against the average bit rate curves for the first 60 frames coded at three different frame rates: 7.5 frames/s, 6 frames/s and 5 frames/s. It is interesting to note that at low bit rates, contrary to normal expectation, the rate-distortion performance at 5 frames/s is almost equivalent to that at 6 frames/s. This is attributed to the failure of motion compensation at low frame rates. Figure 6 presents the original and the reconstructed frame 35 of the Foreman sequence when coded at 6 frames/s for three different bit rates.

This discussion would not be complete without some notes on the rate-distortion performance of our coder when compared against other coders. From the results published for the Foreman sequence by Salembier et al[9], we find that at 5 frames/s and 42Kb/s they achieve a mean PSNR of approximately 30.25 dB for the first 60 frames. They do not explicitly mention how they computed their PSNR results. Our method achieves around 30.0 dB for the luminance alone. If we assume their results to be based on luminance alone, then our method is roughly equivalent to theirs, while if we assume these to incorpo-

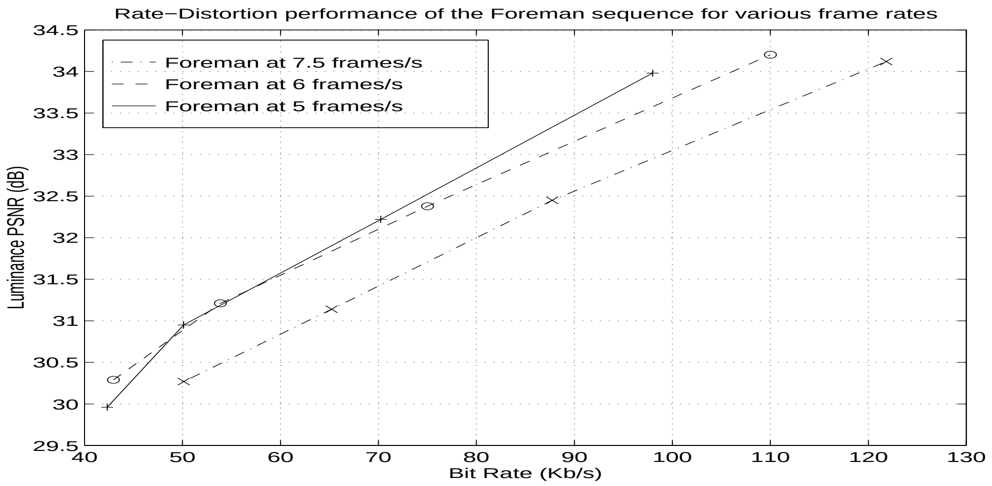
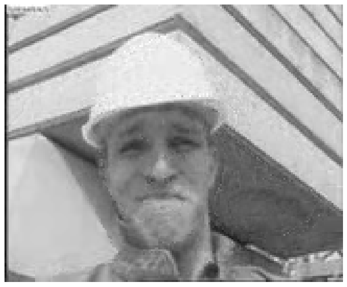
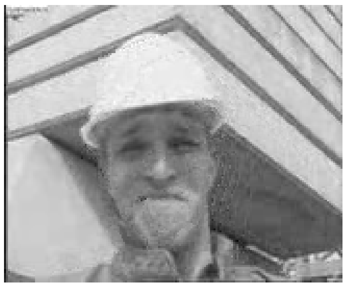


Figure 6. Comparison of Luminance PSNR vs. bit rate for the Foreman sequence coded at three different frame rates: 7.5 frames/s, 6 frames/s and 5 frames/s.

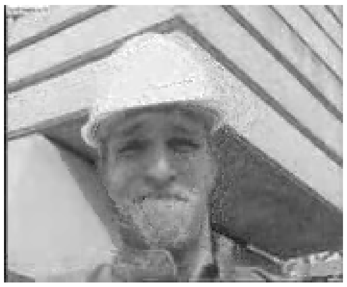
(a) Original Frame 35



(b) Reconstructed Frame 35 (75.0 Kb/s)



(c) Reconstructed Frame 35 (53.9 Kb/s)



(d) Reconstructed Frame 35 (43.0 Kb/s)

Figure 7. Original and Reconstructed Frame 35 of the Foreman sequence coded at 6 frames/s, for three different bit rates: 75.0 Kb/s, 53.9 Kb/s and 43.0 Kb/s.

rate chrominance information as well, ours is superior. Likewise, for the Miss_America sequence, the results published by Yokoyama et al[1], do not explicitly specify how the PSNR is calculated. If we assume that these include both luminance and chrominance components, then our results are superior.

It is to be noted however, that while the rate-distortion efficiency of our coder is superior or equivalent to that reported by Yokoyama et al[1], and that by Salembier et al[9], it still cannot compete with the highly optimized TMN-2.0 implementation of H.263. The H.263 coder is currently about 2 dB better than our region-based coder at the same bitrates. However, we do hope to bridge the gap very soon. Our investigations show that the performance loss when compared against H.263 is in fact a result of inefficiency of the prediction module. More specifically, forward prediction is not a very good option for some of the segmented regions in the previous reconstructed frame. A certain amount of adaptivity in selecting where to use forward prediction and where to use backward prediction will greatly improve the coding performance. On the other hand, the region-based coder, unlike H.263, has the necessary flexibility to incorporate possible object-based functionalities.

Finally, let us mention that some of the techniques in this work have also been applied for developing an object-based video representation system NeTra-V[15].

9. CONCLUSIONS AND FUTURE DIRECTIONS

In this paper we have presented a region-based video coder that eliminates the need to transmit expensive shape-information to the decoder by using a reliable segmentation scheme on the previous reconstructed frame. Forward motion estimation is performed by hierarchical region matching analogous to hierarchical block-matching[13]. As a result of forward prediction, overlapped and uncovered regions arise in the motion compensated frame. The conflicts in the large overlapped regions are resolved by using an Overlapping Distortion matrix. The large uncovered regions are filled by backward motion compensation. For residual error coding, we have used DCT-based coding similar to that used in H.263.

Needless to say, the encoder has ample scope for improvements. First, while a hierarchical approach to region matching greatly reduces the encoding complexity, it is still not suitable for many applications. A better approach is to perform the hierarchical region matching only upto a certain low level of resolution, and then to use the results as the initial guess in a gradient based optimization at full resolution. Second, the way the adjacent regions are currently merged, based on a measure of the similarity in motion, is very arbitrary. While a merge eliminates the possibility of overlapped and uncovered regions between two segments, it also increases the prediction error. The motion parameters for the regions should be so chosen as to minimize jointly the prediction error, and the motion difference with the neighboring segments (which in turn reduces the overlapped and uncovered regions). Third, a region-based residual error coding method will have several advantages. Although this may necessitate use of somewhat arbitrary techniques to estimate the current frame's segmentation map (see Section 7), there will be additional flexibility in allocation of bits to various segments depending on their individual prediction errors and their subjective importance. Moreover, this will pave the way for possible multimode coding of the segments[9]. Individual segments can then be coded in one of several modes, and the coder should make a joint optimization to maximize the rate-distortion performance.

10. ACKNOWLEDGEMENTS

This work was supported by ONR Grant N00014-95-1-1214. The authors also acknowledge W. Y. Ma and B. S. Manjunath for generously providing us with the code for the spatial segmentation algorithm.

11. REFERENCES

1. Y. Yokoyama, Y. Miyamoto, and M. Ohta, "Very low bit rate Video Coding using Arbitrarily Shaped Region-based Motion Compensation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 5, no. 6, Dec. 1995.
2. Y. Yokoyama, Y. Miyamoto, and M. Ohta, "Very low bit rate Video Coding with Object-based Motion Compensation and Orthogonal Transform," *Proceedings of the SPIE, Visual Communication and Image Processing*, vol. 2094, pp. 12-23, Nov. 1993.
3. W. Y. Ma and B. S. Manjunath, "Edge Flow: A Framework of Boundary Detection and Image Segmentation," *Proceedings IEEE International Conference on Computer Vision & Pattern Recognition*, pp. 744-749, June 1997.
4. W. Y. Ma and B. S. Manjunath, "NeTra: A toolbox for navigating large image databases," *Proceedings of IEEE International Conference on Image Processing*, 1997.
5. H. G. Musmann, M. Hotter, and J. Ostermann, "Object-oriented analysis-synthesis coding of moving images," *Signal Processing, Image Communication*, vol. 1, no. 2, pp. 117-138, Oct. 1989.
6. M. Hotter, "Object oriented analysis-synthesis coding based on moving two-dimensional objects," *Signal Processing, Image Communication*, vol. 2, no. 4, pp. 409-428, Dec. 1990.
7. N. Diehl, "Object-oriented motion estimation and segmentation in image sequences," *Signal Processing, Image Communication*, vol. 3, no. 1, pp. 23-56, Feb. 1991.
8. P. Gerken, "Object-based analysis-synthesis coding of image sequences at very low bit rates," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 4, no. 3, pp. 228-235, June 1994.
9. P. Salembier, F. Marques, M. Pardas, J. Ramon, J. Morros, I. Corset, S. Jeannin, L. Bouchard, F. Meyer, and B. Marcotegui, "Segmentation-based Video Coding System allowing manipulation of objects", *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 7, no. 1, Feb. 1997.
10. Special Issue on MPEG-4, *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 7, no. 1, Feb. 1997.
11. T. Sikora, "MPEG-4 Video Standard Verification Model," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 7, no. 1, pp. 19-31, Feb. 1991.
12. F. Pereira and T. Alpert, "MPEG-4 Video Subjective Test Procedures and Results," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 7, no. 1, pp. 32-51, Feb. 1991.
13. M. Bierling, "Displacement estimation by hierarchical block-matching,", *Proceedings of the SPIE, Visual Communications and Image Processing*, vol. 1001, pp. 942-951, Nov. 1988.
14. ITU-T Recommendation H.263, "Video Coding for Low Bit Rate Communication," 1995.
15. Y. Deng, D. Mukherjee, and B.S. Manjunath, "NeTra-V: Towards an object-based video representation", *Proceedings of the SPIE*, vol. 3312, 1998.