ELSEVIER

Signal Processing: *Image Communication* 6 (1994) 357–376

SIGNAL PROCESSING:
IMAGE
COMMUNICATION

Change detection and texture analysis for image sequence coding

Zohar Sivan*, David Malah

Department of Electrical Engineering, Technion-Israel Institute of Technology Haifa, 32000, Israel

Received 23 June 1993; revised 15 February 1994

Abstract

In Hybrid DPCM image sequence coders, every block with high variance in the corresponding difference block is replenished even if it belongs to a stationary background region, as could be the case for coarse textured blocks. We present an improved Hybrid DPCM image sequence coder, which includes a block classification unit. This unit detects stationary textured background blocks and allows copying such blocks instead of replenishing them, thus reducing the bit-rate of the coder. The unit consists of three parts. The first part is a statistical-based change detector. Segmentation of each image into textured and smooth regions, a fast converging deterministic relaxation procedure and a multi-resolution approach are used to obtain the final moving/stationary segmentation. In the second part, a Gaussian AR model-based texture matching test is proposed. The third part detects edges in stationary blocks. To avoid edge related artifacts, blocks that contain an edge are unconditionally coded. The coder which incorporates the proposed change detector was found, in simulations, to provide a substantial reduction in bit-rate, while maintaining the quality of the reconstructed sequences, in coding image sequences which contain large areas of coarse textured background.

Key words: Image sequence coding; Video coding; DPCM; Change detection; ICM; Multi-resolution; Image segmentation; Texture; Edge detection; Gibbs distribution; AR modeling

1. Introduction

In standard image sequence coders such as H.261 [1, 2], the current frame is predicted from the previous reconstructed frame using local motion compensation, and the prediction error is coded. Such coders do not take into special consideration the situation where the images contain coarse random textured background regions. Such regions may cause large amplitude fluctuations in the difference frame, because of jitter (both in the camera position and in the sampling process), although no change is detected by the human observer. Detecting stationary textured background blocks and copying such blocks from the previous frame instead of coding them can substantially reduce the bit-rate of the coder while maintaining good quality of the reconstructed frames.

One possible method for performing this detection is suggested by Papathomas and Malah in [22]. According to their method, the detection of stationary textured blocks is performed using a decision tree, based on thresholds obtained from

* Corresponding author. Present address: IBM Israel, Science & Technology, MATAM – Advanced Technology Center, Haifa 31905, Israel.

psychophysical experiments. The tests in the decision tree include the evaluation of the mean and variance of the difference block with and without local motion compensation, and the comparison of these values to the above-mentioned thresholds. Blocks that are considered candidates of belonging to stationary textured background regions undergo various verification tests before being labeled as such, in which case they are actually copied from the previous reconstructed frame.

This method has the disadvantage that some of the verification tests are heuristic in nature and are performed on the difference between original and *reconstructed* frames, so that quantization errors may influence the results.

In this paper, a different approach is presented. This approach is based on statistical models for detecting stationary textured blocks, with the classification task being performed in several steps. The first step is change detection. The segmentation of images into changed and unchanged regions can help in more accurate calculation of motion parameters [11]. It also improves the *selective* coding of information, by copying background blocks that normally would be coded because of high variance in the difference block, and thus substantially reduces the bit-rate of the coder [15]. In addition, this approach is superior to the method of classifying moving and stationary regions according to motion vectors (which are usually obtained by block matching). Typical problems in using motion vectors for such classification are, on one hand, stray motion (particularly in textured background regions), and on the other, no apparent motion if the actual motion is beyond the search region of the block matching algorithm, or when a new object appears in the scene. Thus, motion vectors may not be reliable enough for the classification task.

The simplest change detector is obtained by thresholding the difference of two consecutive frames. Pixels with absolute value above a certain threshold are considered as moving, whereas the others are considered as stationary [10]. The results obtained by this method are rather poor when the images contain wide-band coarse texture. This is because of the jitter mentioned above. Hence, pixels which belong to such stationary textured regions might be considered as moving. On the other hand, using low-pass filtering, as in [10], may cause oversmoothing of the difference frame.

Another approach for change detection is suggested by Karmann et al. [16]. They use Kalman filtering of certain reference frames in order to adapt to changing image characteristics. The adaptation, however, is much too slow for image sequence coding, where adaptation within each frame is required.

A different class of algorithms is the class of statistical model-based algorithms, to which our algorithm belongs. Such an approach appears to help in alleviating some of the problems characterizing the threshold-based approach. This is mainly because spatial interaction between neighboring pixels can be taken into account. Cafforio and Rocca [9] assume that pixels in the difference frame are statistically independent and each pixel is a zero-mean Laplacian stochastic variable. The difference between moving and stationary pixels is reflected in the value of the standard deviation σ for each type. The segmentation field was modeled as a first order Markov chain along rows and the solution was obtained by solving a Maximum Aposteriory Probability (MAP) problem using the Viterbi algorithm.

An extension of this model into a bi-dimensional second order causal Markov chain is suggested by Mori et al. [20] and also by Driessen et al. [11]. The two methods differ only in the Probability Distribution Function (PDF) assumed for the pixels in the difference frame. In both methods transition probabilities have to be empirically obtained.

A more sophisticated model is suggested by Lalande and Bouthemy [18]. They use a spatio-temporal Markov Random Field (MRF) model through Gibbs distribution and construct a cost function which is minimized using a deterministic iterative algorithm. Their method is intended for traffic control and hence is not directly applicable to coding applications. This is because in their scheme, uncovered background is considered to be a stationary region and hence will not be coded.

All the above methods use deterministic relaxation algorithms that are very sensitive to the initial segmentation. In addition, no consideration is given to the characteristics of the images, which may assist in a more accurate change detection (see Section 2.1).

Since after a block is copied from the previous reconstructed frame, possible errors cannot be corrected, more tests are needed to ensure that the decision to copy the block is correct. One possible test is to check if the textures in the two stationary blocks, in the current and previous original frames, are similar enough. Another test needed, is to detect edges in stationary blocks, since the Human Visual System (HVS) is very sensitive to edges. Blocks with edges should be unconditionally coded, in order to avoid possible artifacts in the reconstructed frames.

The proposed change-detection algorithm is an extension of a previous algorithm that we suggest in [24]. The algorithm uses a Markov Random Field (MRF) model, through Gibbs distribution, to describe the segmentation field. A Laplacian distribution function is used to model the difference frame. A cost function, based on these distribution functions, is constructed and a Maximum Aposteriory Probability (MAP) problem is solved. The algorithm has several advantages over previous algorithms which use similar techniques [9, 11, 15, 18, 20]. The first advantage is adaptation to image characteristics, which is achieved by segmenting each image into *smooth* and *textured* regions. The second one is that the strong dependence of the deterministic relaxation algorithm on the initial segmentation [7] is reduced by performing *several* iterations of the basic minimization procedure, where each one is initialized by a segmentation field based on the parameters obtained in the previous iteration. The result with the lowest cost-function value is chosen to be the final segmentation.

The segmentation results are further improved by using a multi-resolution approach. This approach is used in image coding applications [8] and also in texture segmentation [7]. In segmentation applications, this approach enables improved results when used within an iterative minimization process. A substantial reduction in the computational cost of the minimization process is also achieved, since most of the operations are performed on a coarse resolution image which contains less pixels than the original image at a higher resolution.

The texture matching test and the detection of edges in stationary blocks, are based on a method for cluster validation suggested in [7]. This method uses statistical models of the data and the similarity of the blocks is evaluated via the Akaike Information Criterion (AIC) measure [3].

Details of the texture matching test and the edge detection are given in Sections 3 and 4, respectively, after the description of the change-detection algorithm in Section 2. A description of the block classification unit is given in Section 5 and simulation results are presented in Section 6. Conclusions are drawn in Section 7.

2. Change-detection algorithm

The algorithm can be separated into two main parts. In the first part, a Texture/Smooth Segmentation (TSS) of the image is performed. In the second part, a MAP problem is solved by minimizing a cost function which takes into account the TSS. The minimization is performed using a multi-resolution approach, i.e., applying it on several resolution levels of the images, as will be explained in Section 2.3.3.

2.1. Texture/smooth segmentation

The motivation for this segmentation is the fact that pixels in the *difference frame* belonging to *smooth moving* regions may have a *smaller* standard deviation than pixels belonging to *textured stationary* regions and thus, erroneously, smooth moving regions may be considered as stationary, or textured stationary regions may be considered as having been changed. The TSS allows the characteristics of each region to be taken into consideration in the change detection process. In this context, textured regions can be regarded as *non-smooth* regions and as such may contain edges, lines, etc.

The TSS is performed using blocks of size 16×16 pixels. Each original image block is modeled as a causal Gaussian Auto Regressive (AR) process [7] with four coefficients. These coefficients relate each pixel in the block with four of its nearest neighbors. The choice of the block size is a compromise between good statistical behavior of the AR model (preferring larger blocks) and accuracy of the segmentation (preferring smaller blocks). If we define $v_{i,j}$ as the value of the (i,j) pixel in the block, after subtracting the block's average, we get that the prediction error of the (i,j) pixel, $\tilde{v}_{i,j}$, is given by

$$\tilde{v}_{i,j} = v_{i,j} + a_1 v_{i,j-1} + a_2 v_{i-1,j+1} + a_3 v_{i-1,j} + a_4 v_{i-1,j-1}, \tag{1}$$

where a_i are the AR model coefficients.

Let V be a vector containing all the M pixels in the block arranged according to a *raster scan* of the block (starting at the top left corner of the block and moving along rows), and let $\tilde{V}$ be the vector containing all the prediction errors in the block arranged in the same order as in V. In matrix form we can write $\tilde{V} = AV$, where A is an $M \times M$ lower triangular matrix, which contains the AR model coefficients. For demonstration, A is shown below for a block of size 4×4 (i.e. $M = 16$):

$$A = \begin{bmatrix}
1 & 0 & .0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
a_1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & a_1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & a_1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
a_3 & a_2 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
a_4 & a_3 & a_2 & 0 & a_1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & a_4 & a_3 & a_2 & 0 & a_1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & a_4 & a_3 & 0 & 0 & a_1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & a_3 & a_2 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & a_4 & a_3 & a_2 & 0 & a_1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & a_4 & a_3 & a_2 & 0 & a_1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & a_4 & a_3 & 0 & 0 & a_1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a_3 & a_2 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a_4 & a_3 & a_2 & 0 & a_1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a_4 & a_3 & a_2 & 0 & a_1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a_4 & a_3 & 0 & 0 & a_1 & 1
\end{bmatrix}. \tag{2}$$

It is observed from (2), that A has an all 1's main diagonal and it is clear that its determinant equals to 1. Using these characteristics of A and the assumption that the elements in $\tilde{V}$ are a sample from an i.i.d. Gaussian process, the probability density function of V can be shown [7] to be

$$p_V(V) = \sqrt{\det(AA^{\mathrm{T}})}\, p_{\tilde{V}}(\tilde{V}) = \left(\frac{1}{\sqrt{2\pi\tilde{\sigma}^2}}\right)^M \exp\left\{-\frac{1}{2}\sum_{i,j\in B}\frac{\tilde{v}_{i,j}^2}{\tilde{\sigma}^2}\right\}, \tag{3}$$

where M is the number of the pixels in the block B, $\det(AA^{\mathrm{T}})$ is the determinant of AA^{T} (which equals here to 1) and $\tilde{\sigma}^2$ is the variance of the prediction error in the block. $p_{\tilde{V}}(\tilde{V})$ is the probability density function of the prediction error vector $\tilde{V}$.

The parameters of the model (a_i, $i = 1,2,3,4$; $\tilde{\sigma}^2$) are estimated using Maximum Likelihood Estimation (MLE) [7], where the pixels outside the block are considered to be with zero value. The TSS is performed using an empirically obtained threshold T_{A}. The threshold can be made adaptive by using, for example, the distribution of the block variances in each region type (textured or smooth) in the previous frame. If the variance of the *prediction error* of a block in each of the two consecutive frames is below T_{A}, then the corresponding block in the difference frame is considered to be *smooth*, otherwise it is considered non-smooth, i.e., *textured*. The resulting TSS map is denoted as S. Following the TSS, each region type may contain both moving and stationary pixels.

2.2. Statistical models

Two statistical models are used in the second part of the change-detection algorithm. Let Y denote the difference frame and let X denote the Moving/Stationary Segmentation (MSS) field.

We use a Laplacian Probability Distribution Function (LPDF) to describe the statistical behavior of the pixels in Y. We assume that the difference pixels are statistically independent. The conditional probability density function of the difference pixel $y \in Y$ is given by

$$p(Y=y \mid X=x,\, S=s) = \frac{1}{\sqrt{2}\sigma_{x,s}} \exp\left\{\sqrt{2}\frac{|y|}{\sigma_{x,s}}\right\}. \tag{4}$$

The four possible standard deviation values $\sigma_{x,s}$ of each difference pixel depends on its being moving or stationary and its assignment to either a smooth or a textured region.

The MSS field X is modeled as an MRF with a second order neighborhood system [7, 18]. To describe $p(X)$ we use a Gibbs distribution, where only two pixel cliques are considered in order to reduce the number of necessary parameters. We distinguish between diagonal cliques and vertical or horizontal ones, as done in [7]. Using the local characteristics of the MRF, $p(X)$ is given by

$$\begin{aligned} p(X) &= \frac{1}{Z}\exp\{-U(X)\} \\ &= \frac{1}{Z}\exp\left\{-\sum_{\text{all cliques } c} V^c(x_c)\right\} \\ &= \frac{1}{Z}\exp\{-(\beta_{1,s}t_1 + \beta_{2,s}t_2)\}. \end{aligned} \tag{5}$$

$U(X)$ is the energy function which is defined as the sum of local potentials $V^c(x_c)$ and Z is a normalization constant. t_1 is the *total* number of vertical/horizontal cliques and t_2 is the *total* number of diagonal cliques, which contain each, two *differently labeled* pixels. $\beta_{1,s}$ and $\beta_{2,s}$ are the parameters of the Gibbs distribution function and satisfy the relation $\beta_{1,s} = \sqrt{2}\beta_{2,s}$ $(\beta_{1,s}, \beta_{2,s} > 0)$. *Different* sets of parameters are assigned to textured and smooth regions. These parameters control the effect of the spatial interaction between a pixel and the other pixels in its neighborhood on $p(X)$.

Our problem is to find the best MSS field X given the data in the difference frame Y and the TSS field S. Using the MAP criterion, Bayes' formula, and the fact that X is statistically independent of S, we get

$$\begin{aligned} X_{\text{map}} &= \arg\max_X p(X \mid Y, S) \\ &= \arg\max_X \left\{\frac{p(Y \mid X, S)p(X)}{p(Y \mid S)}\right\}. \end{aligned} \tag{6}$$

Taking into account the assumption that $p(Y \mid S)$ is independent of X, we get

$$X_{\text{map}} = \arg\min_X\{-\ln[p(Y \mid X, S)p(X)]\}. \tag{7}$$

The desired segmentation field is obtained by minimizing the above cost function. From (4)–(7) we obtain

$$X_{\text{map}} = \arg\min_X \left\{\sqrt{2}\sum_{i,j}\left[\frac{|y_{i,j}|}{\sigma_{x,s}} + \ln\sigma_{x,s}\right] + \beta_{1,s}t_1 + \beta_{2,s}t_2\right\}. \tag{8}$$

Note that increasing the value of $\beta_{1,s}$ or $\beta_{2,s}$ results in giving more weight to spatial interaction in the MSS field X than to the magnitude of pixels in the difference frame Y.

2.3. Optimization procedure

A global minimum can be obtained by minimizing over the entire segmentation field using the method known as simulated annealing [14]. Since this method is not practical in our application, we use the Iterated Conditional Modes (ICM) minimization algorithm proposed by Besag [6] and applied in [7, 15] as well. The basic idea behind the ICM is to replace pixels in the segmentation field X with a value which maximizes the conditional density function *at each pixel*, while fixing the remaining pixels in X. Each time a replacement is performed, the cost function is assured to decrease (or at least not to increase). This deterministic relaxation algorithm requires a relatively small amount of computations and converges fast, but may converge to a local minimum. We apply the following two-level iterative procedure.

2.3.1. *Inner-loop procedure*

Beginning with an initial MSS field X_{init}, which is calculated in the outer-loop procedure (see Section 2.3.2 below), we minimize the cost-function in (8) using the ICM method, until no change occurs in X. The updating of X can be done pixel by pixel in a raster scan, but the result is found to be sensitive to the scan mode [7]. Therefore, X is divided into several groups, each containing pixels that are *independent* of each other, given the pixels in the remaining groups. When such a division is applied to a field defined with a second order neighborhood system, four different groups are created. In Fig. 1, the pixels are divided among the four groups which are denoted by K, O, U, and V. Pixels which belong to the same group do not belong to the same neighborhood and thus are statistically independent given the pixels in the remaining groups.

This division of X reduces the sensitivity to the scan mode and allows parallel computation since the pixels in each group are independent of each other. A similar division is used in [7, 15].

After convergence is achieved, the $\sigma_{x,s}$ values of the LPDF in (4), for each of the four segment types ('moving texture', 'moving smooth', 'stationary texture' and 'stationary smooth'), are reestimated using MLE as in the right-hand side of (10) below. The minimization step is repeated, with the new σ's and the last obtained X as an initial segmentation, until the change in the $\sigma_{x,s}$ values is below a certain threshold. This is actually a simultaneous minimization and estimation procedure but different than the one suggested by Besag [6], by the fact that according to Besag's method the estimation should be performed after each ICM cycle, whereas we perform the estimation after the ICM process is completed. This estimation procedure was found to give a lower cost-function value than Besag's method for the tested sequences, at a lower computational cost. During this procedure the cost function is assured to decrease.

```
K O K O K O K O
U V U V U V U V
K O K O K O K O
U V U V U V U V
```

Fig. 1. Division of the field X into four groups (K, O, U, V) of independent pixels.

2.3.2. *Outer-loop procedure*

In the outer-loop procedure, the inner-loop process is performed several times, each time with a different initial segmentation X_{init}. Each X_{init} is obtained by MLE of X using only the information in the difference frame, the segmentation field S and the $\sigma_{x,s}$ values. X_{init} is given then, pixel by pixel, by

$$(X_{\mathrm{init}})_{i,j} = \arg\max_{x} \{p(Y = y \mid X = x, S = s)\} = \arg\min_{x} \left\{\sqrt{2}\left[\frac{|y_{i,j}|}{\sigma_{x,s}} + \ln \sigma_{x,s}\right]\right\}. \quad (9)$$

An exception is the first time the inner-loop process is performed, because the initial σ's are yet unknown. At that time these are obtained by thresholding the difference frame Y, to obtain a segmentation field X_0, and performing an MLE of $\sigma_{x,s}$ for each of the four possible segment types according to

$$\sigma_{\mathrm{mle}} = \max_{\sigma} \{p(Y \mid X_0, S, \sigma)\} = \frac{\sqrt{2}}{\|W\|} \sum_{y \in W} |y|, \quad (10)$$

where W denotes the segment type and $\|W\|$ is the number of the pixels in that segment.

In determining X_0 we use different thresholds for the smooth and textured segments, since the characteristics of these segments are different.

Each outer-loop iteration consists of one complete inner-loop process. The cost-function value at the end of each outer-loop iteration is evaluated and the iterations continue until this value stops to decrease. The segmentation field X corresponding to the lowest cost-function value obtained in this process is the final segmentation.

The use of several initial segmentations reduces the effect of the value of the initial threshold on the final segmentation result, thus making it more robust.

2.3.3. *Multi-resolution approach*

The idea behind the multi-resolution approach is to perform the segmentation algorithm at several resolution levels, starting with the coarsest resolution. The segmentation obtained at each resolution level serves as the initial segmentation for the next level. This way, the segmentation at each resolution level 'guides' the next level.

Each resolution level corresponds to a particular level in a quad tree, so that a pixel at a given resolution level corresponds to four pixels at the next finer resolution level as is shown in Fig. 2.

Since the algorithm is based on statistical models, we are interested in the statistical behavior at different resolution levels. Using the fact that the pixels of the difference frame Y are assumed to be independent, we can generate a difference pixel at a certain resolution level by summing the four appropriate difference pixels in the finer resolution level. This summation is equivalent to low-pass filtering, which is typically required in every decimation process.

Using this structure, the independence of the difference pixels is maintained in each of the resolution levels.

The *same* minimization method is applied to all the resolution levels. This implies that we must assume that the distribution of the pixels at each resolution level can be approximated by a Laplacian distribution. Since by the central limit theorem, the distribution of the sum of N i.i.d. variables tends to be Gaussian as N tends to infinity, the assumption of Laplacian distribution is certainly not valid when we move to very low resolution levels. Actually, according to our experiments, the Laplacian approximation is valid only for two resolution levels, i.e., the original resolution level and the next lower one. However, the contribution of using even one additional resolution level is considerable as is shown in Section 5.

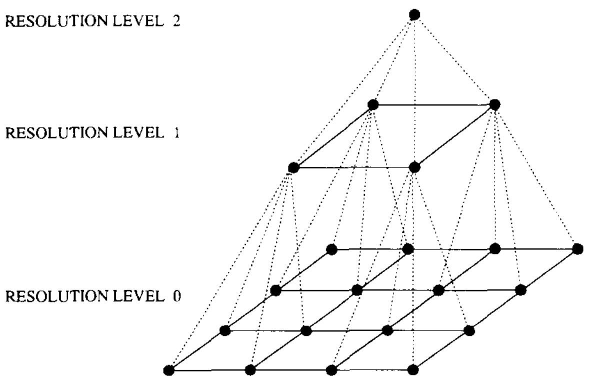

Fig. 2. Schematic description of the multi-resolution pyramid.

After the segmentation process is completed at a certain resolution level, the resulting segmentation field X is projected to the finer resolution level by copying its label to its four generators at the finer resolution level.

The TSS is performed at the finest resolution level only. The classification of a pixel at a certain resolution level into smooth or textured region is then done by checking the classification of *all* its generators at the finest resolution level. Since this segmentation is performed using blocks of size 16×16 pixels, using even four resolution levels in addition to the original resolution, ensures that the generators of a pixel at the coarsest resolution level are classified in the same way (this is because at the fifth resolution level a pixel is generated by summing the 16×16 corresponding pixels at the original resolution level). Since we use only two resolution levels, the classification of a pixel at the coarser resolution level as belonging to a smooth or textured region is uniquely determined.

Since the final segmentation at each resolution level 'guides' the next level, by serving as its initial segmentation field, the outer-loop procedure is performed only at the *lowest* resolution. At other resolution levels only the inner-loop procedure is performed. For this reason, a large part of the computations is performed only at the lowest resolution level, in which the number of pixels is small, as compared with the original resolution level, thus reducing the overall computational cost.

2.3.4. *Summary of the change-detection algorithm*

General notes:

- The proposed change-detection algorithm uses 2 resolution levels (level 0 indicates the finer resolution and level 1 the coarser one).

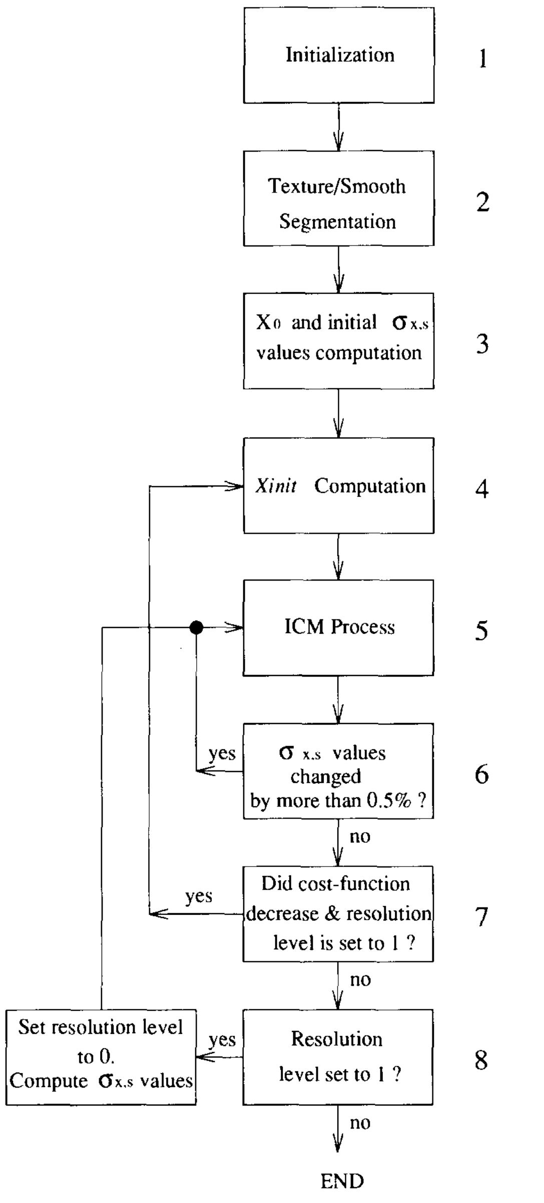
Initialization
1
Texture/Smooth Segmentation
2
X0 and initial σx,s values computation
3
Xinit Computation
4
ICM Process
5
σ x,s values changed by more than 0.5% ?
6
yes
no
Did cost-function decrease & resolution level is set to 1 ?
7
yes
no
Resolution level set to 1 ?
8
yes
Set resolution level to 0. Compute σx,s values
no
END

Fig. 3. Schematic diagram of the proposed change-detection algorithm.

- The outer-loop iteration is performed at least twice, to make sure that the cost-function value stopped decreasing. It is performed *only* at the coarser resolution level (level 1).
- The texture/smooth segmentation is performed *only* at the finer resolution level (level 0).
- The initial segmentation X_0 is computed using a simple thresholding of the frame-difference using different threshold values for smooth and textured regions.

The proposed algorithm is summarized below and in the block diagram of Fig. 3.

1. Initialization: Set the initial cost-function value to a very large number (e.g., 10^{10}) and set the resolution level to 1.
2. Perform the texture/smooth segmentation on the current frame and classify each block to be either smooth or textured, determining the segmentation field S.
3. Compute the frame-difference for resolution level 1 by first transforming the current and previous frames to resolution level 1, as described in Section 2.3.3, and then subtracting the current frame from the previous one, to obtain the necessary frame-difference at level 1. Perform thresholding on this frame-difference (using different threshold values for smooth and textured regions), to obtain X_0. Calculate initial $\sigma_{x,s}$ values using (10) and X_0.
4. Outer-loop iteration: Perform MLE of X using the estimated $\sigma_{x,s}$ values to obtain X_{init} according to (9).
5. Inner-loop iteration: Find X_{map} according to (8), using the ICM minimization algorithm. The frame-difference is the data used in the minimization process.
6. Estimate new $\sigma_{x,s}$ values using (10). If they change by more than a given amount (e.g., 0.5%), end the current inner-loop iteration and return to step 5. Else, go to step 7.
7. Evaluate the cost-function in (8). If the resolution level is 1 and the evaluated cost-function value is smaller than the previous value, end the current outer-loop iteration and return to step 4. Else, go to step 8 and the final segmentation field X at the *current resolution level* is the one obtained in the previous outer-loop iteration, corresponding to the lowest cost-function value.
8. If the resolution level is 1: Project the last obtained segmentation field X on resolution level 0 as described in Section 2.2.3, set the resolution level to level 0, calculate $\sigma_{x,s}$ values using (10) and return to step 5.
   If the resolution level is already 0: END.

3. Texture-matching test

This test is intended for increasing the probability of correct block classification. When the image sequence contains a moving random-textured region on a textured background with different texture features, the change detector may classify such a moving region as stationary, because the difference frame contains similar amplitude fluctuations as in the case of stationary coarse textured background. To avoid misclassification in such a case, a test for texture similarity is needed.

The texture-matching test is based on the procedure for texture segmentation proposed in [7] with some modifications. In [7], the Akaike Information Criterion (AIC) is used to decide whether two regions are similar enough, so they can be combined into one. The analogy to the texture-matching problem is straightforward. We wish to test if two corresponding blocks contain similar textures, so that the block in the current reconstructed frame can be copied from the previous reconstructed frame. If the blocks contain similar textures, then, in a hypothetical segmentation procedure, they could have been combined into one region. Using this terminology, the texture-matching test can be performed in the same way as is done in [7]. We evaluate the AIC for the case in which the two blocks are assumed different and for the case in which the blocks are assumed similar (in this case they are combined into one bigger block). The smaller AIC value of the two determines the result of the texture matching test.

The AIC is given by

$$\mathrm{AIC}(k) = -2\ln p\{\text{Maximum Likelihood}(k)\} + 2k', \tag{11}$$

where k is a running index explained below, $p\{\text{Maximum Likelihood}(k)\}$ denotes the probability density function (PDF) of the data samples, k' is the number of *identifiable* parameters used in the model which describes the data and N is the number of data samples. The parameters of this model are estimated using MLE.

One problem in using the AIC is its statistical behavior. Kashyap proved in [17] that the AIC estimator is not consistent statistically. He also proposed a modification of the AIC in order to make it consistent. The Modified AIC (denoted MAIC) is given by

$$\mathrm{MAIC}(k) = -2\ln p\{\text{Maximum Likelihood}(k)\} + k'\ln(N). \tag{12}$$

When the AIC or MAIC are used for the determination of AR model order, k indicates a possible order for the AR model ($0 < k \leqslant K_0$, where K_0 is an upper limit for the AR model order). k' equals k in this case.

In the texture-matching test we use $k = 1$ for the case where the two blocks are similar enough (they can be considered as *one* bigger block). In this case $k' = 6$ as it includes the four AR model coefficients, the prediction error variance and the mean of the block. For the case where the two blocks are considered as two different blocks, $k = 2$ and k' includes now two sets of AR model coefficients, two prediction error variances, two mean values and a weighing coefficient, i.e. $k' = 13$. The weighting coefficient for $k = 2$ is used in the calculation of the PDF. This PDF is a *weighted* combination of the PDFs of the two blocks [7, 19]. Since the sum of the two weighting coefficients should be 1, they are dependent, and only one should be taken into account.

To construct the PDF of the data samples we use the Gaussian AR model described in Section 2.1. The estimation of the model parameters is performed using MLE in the same way as in [7].

4. Detection of edges in a block

As stated before, the HVS is very sensitive to edges. Very small movements of edges in the frames may not be detected by both the change detector and the texture-matching test. If a stationary block which contains an edge is continuously copied, such movements can accumulate and eventually be detected, resulting in the coding of the block. This update of the block may be perceived as a sudden change, to which the HVS is very sensitive. To avoid such a situation, it was decided that every block that contains an edge should be *unconditionally* coded. An algorithm for the detection of

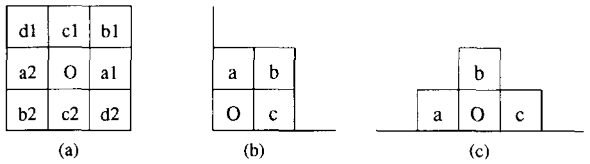

Fig. 4. Description of the pairs of blocks tested in the first sub-test for edges in block O. (a) Blocks inside the image. (b) Blocks in a corner of the image. (c) Blocks on the border of the image.

edges in a block is therefore needed. In this section a method for detecting edges in a block is presented.

The proposed method is composed of two sub-tests. The first level is based on the method suggested by Fan [13]. The idea is that if a block contains an edge which separates two different regions, then blocks from different sides of the tested block will be different in nature. We therefore regard the problem of detecting an edge in a block as a similarity test between two blocks and hence, the prediction error variances from the texture matching test can be used. In this way, the computational cost of the test is reduced. If we check all four possible pairs of blocks for similarity, the existence of an edge will be reflected in the results of the similarity test. In Fig. 4(a) the four pairs (a1, a2), (b1, b2), (c1, c2), (d1, d2) and the tested block O are shown. For border and corner blocks this test cannot be performed as stated. Here, the similarity test is performed between the tested block O and its three closest neighbors a, b and c, as is shown in Fig. 4(b) for corner blocks and in Fig. 4(c) for border blocks.

The similarity of the blocks is evaluated via the MAIC measure using the texture parameters calculated in the texture-matching test with one difference. The average of the block is not subtracted from the pixel's value. This is because a difference in the DC value of the two blocks is relevant to the similarity test in this case. If at least one pair contains different blocks, the block O in Fig. 4 is suspected of containing an edge and the second sub-test is applied. The need for a second test arises from the fact that blocks *near* an edge are also declared as containing an edge [13, 23].

The second sub-test is also based on the idea of comparing two blocks to check similarity, but the nature of the parameter estimation of the AR model for the blocks is somewhat different from the first sub-test (in which the parameters estimated during the texture-matching test are used). Since the main problem in the first sub-test is the coarse resolution of the test (blocks are of size 16×16), a reasonable way to overcome the above-mentioned problem is to perform the test at a finer resolution, i.e., using smaller blocks.

The second sub-test is therefore considered as a homogeneity test of the block. Each tested block is divided into four sub-blocks of equal size, which are compared to each other to test their similarity. However, since the sub-blocks are of smaller size (8×8), estimating the AR model parameters using the MLE method proposed in [7], gives unsatisfactory results.

For the estimation process, only pixels for which all the needed information exists *within* the block are considered (so that boundary pixels are not used). In addition, we use Least Squares (LS) estimation instead of MLE, because the probability function of the block cannot be written as in (3). This is because $\det(\sqrt{AA^{\mathrm{T}}})$ is not equal to 1 anymore.

The similarity testing itself is now carried out using the Itakura distance measure, which is used mostly in speech but also in images [21]. The Itakura distance D_{I_1} between block 1 and block 2 is given by

$$D_{\mathrm{I}_1} = \ln\left[\frac{\hat{\boldsymbol{a}}_2 R_1 \hat{\boldsymbol{a}}_2^{\mathrm{T}}}{\hat{\boldsymbol{a}}_1 R_1 \hat{\boldsymbol{a}}_1^{\mathrm{T}}}\right] \geqslant 0, \tag{13}$$

where $\hat{\boldsymbol{a}}_1$ and $\hat{\boldsymbol{a}}_2$ are the AR parameters vector of blocks 1 and 2, respectively, and R_1 is the autocorrelation matrix of block 1.

This distance measure is not symmetric, so both D_{I_1} and D_{I_2} are calculated. In addition the distance between the blocks is calculated according to another simpler measure defined by

$$D = \left|\ln\left[\frac{\tilde{\sigma}_1^2}{\tilde{\sigma}_2^2}\right]\right|, \tag{14}$$

where $\tilde{\sigma}_1^2$ and $\tilde{\sigma}_2^2$ are the prediction error variance of blocks 1 and 2, respectively. This distance measure is symmetric.

The final similarity measure used, is calculated using the previous three distance measures according to

$$D_T = \max\{D_{I_1}, D_{I_2}, D\}. \qquad (15)$$

D_T was chosen in this way, because we wish to emphasize the difference between the blocks, making sure that if an edge is present it would be detected. The value of D_T is compared with an empirically obtained threshold T_B, to decide if the blocks are sufficiently similar. T_B can be made adaptive in a similar way to T_A in Section 2.1.

To reduce the computational cost, only stationary blocks classified as containing an edge in the *previous* frame and their eight nearest neighbors are checked for edges, in addition to blocks which were classified as moving in the *previous* frame, but are classified as stationary in the current frame. The corresponding eight neighboring blocks are examined because very small movements of thin edges may not be detected by the change detector. Hence, an edge near the block boundary may move to a neighboring block.

5. Block classification unit

In Fig. 5, a block diagram of the block classification unit is shown. In the first stage the change detection is performed at the pixel level. In the second stage the results are evaluated and if a block contains enough moving pixels (e.g., 10% of the pixels in the block), it is sent to the coding unit. To avoid the possibility of a moving region which is large enough, but divided between adjacent blocks such that the total number of moving pixels in each block is less than the 10% requirement for coding, another test is performed. If the number of moving pixels in a block is less than 10% but more than 5% of the total number of pixels in a block, the boundaries of the block are tested for motion continuity with respect to its four closest neighboring blocks. The test is performed using the moving/stationary segmentation field. If motion continuity exist, the block is sent to the coding unit. Otherwise it is processed by the next stage of the block classification unit. This motion continuity relates to the case where a moving region spans over two or more neighboring blocks, such that each block contains only a part of the moving region.

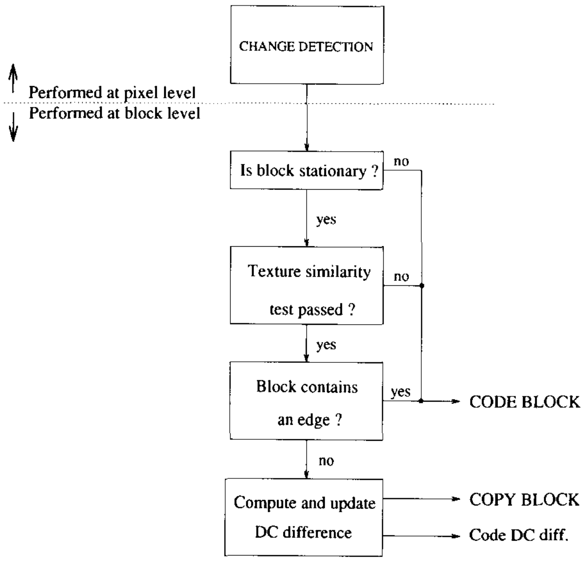

Fig. 5. Schematic description of the block classification unit.

In the third stage, corresponding stationary blocks in consecutive frames are tested for texture matching and if they are not similar, the relevant block is sent to the coding unit. If the blocks are similar, the block in the current frame is checked for edges in the fourth stage. If it contains an edge, it is again sent to the coding unit. In the fifth stage the blocks which are to be copied are checked for DC difference with respect to the corresponding block in the previous frame. The DC level is updated and coded if necessary.

The described classification unit is for scenes without *global* motion, since all the tests check for *local* changes only. If the coder is intended for coding also scenes with global motion (e.g., zoom and pan), a Global Motion Compensation (GMC) unit should be incorporated within the coder. We used such a unit, which operates according to the algorithm suggested in [4, 12]. The GMC algorithm for 3-D scenes suggested in [4, 5] requires segmentation of the scene according to the global motion parameters of each region in the scene. Since at present it provides block-based segmentation, it is not accurate enough for the proposed coder. It could be incorporated within the proposed coder, for coding 3-D scenes with global motion, once the segmentation will be at the pixel level.

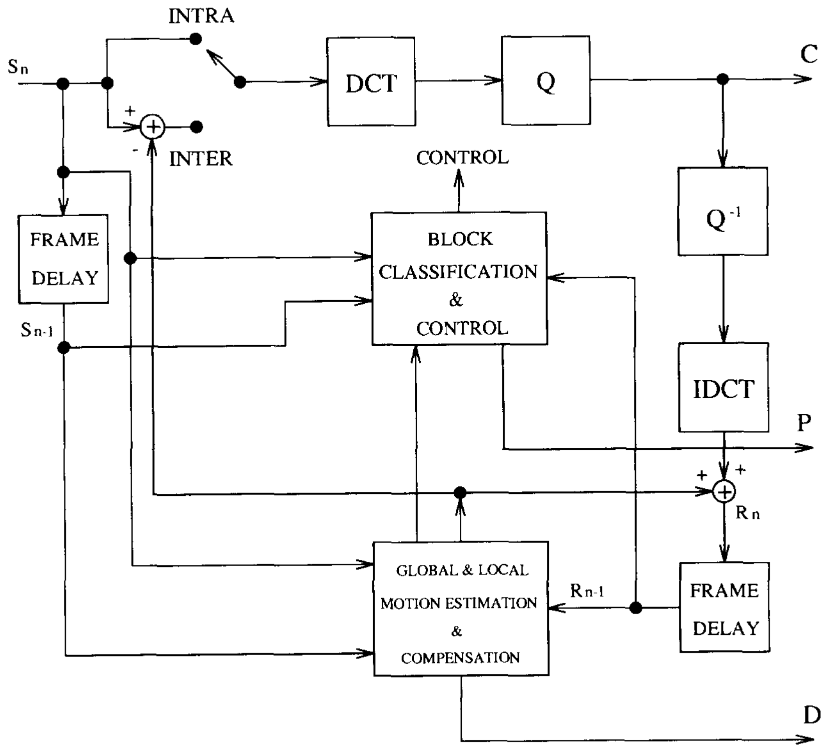
INTRA
Sn
DCT
Q
C
+
-
INTER
CONTROL
FRAME DELAY
BLOCK CLASSIFICATION & CONTROL
Q-1
Sn-1
IDCT
P
+
+
Rn
GLOBAL & LOCAL MOTION ESTIMATION & COMPENSATION
Rn-1
FRAME DELAY
D

Fig. 6. Block diagram of the proposed coder with the block classification unit and the GMC unit. C denotes the quantized DCT coefficients, P the classification data and D denotes the local and global motion parameters.

Another problem associated with GMC is the accuracy of the estimated global motion parameters. Since the Global Motion Parameters Estimation (GMPE) is performed using only two frames at a time, the accumulation of errors is inevitable when a forced copying of blocks is performed in scenes with GMC, as is the case in the proposed coder. A possible solution to this problem is to perform the block classification using a special *reference frame*. The *reference frame* is then updated according to the block classification-unit decisions. The initial reference frame is identical to the first frame in the coded sequence. The updating is performed by copying blocks from the *current* original frame if the corresponding blocks were coded and leaving the other blocks unchanged. In this way, only the relevant errors are taken into account, since when a block is coded the effect of GMPE error on that block can be assumed to vanish. This reference frame replaces the previous original frame in the *next* GMC process. There is almost no computational cost involved in using this method but an additional frame storage for storing the *reference frame* is needed.

A block diagram of the improved coder, which incorporates the block classification unit and a GMC unit is shown in Fig. 6. In this figure, C denotes the DCT coefficients, D the local and global motion parameters and P denotes the block classification information. The information denoted by P is similar to the macro-block mode information used in H.261 [2].

6. Simulation results

We performed three kinds of simulations. The first one concerns the change detector. In this simulation, both synthetic and real data were considered. The next two simulations are concerned with the complete proposed coder. We checked the performance of the proposed coder for a scene which contains local motion only and also for a scene with global motion ('zoom-out'). The coding results were obtained using the coder shown in Fig. 6.

6.1. Change-detector performance

In all the following simulations, the values of β_1 in (5) were chosen empirically to be 1.8 for textured and 1.0 for smooth regions. The value of the threshold for the texture/smooth segmentation was chosen to be $T_A = 15$.

6.1.1. Simulation results for synthetic data

In Fig. 7, simulation results based on synthetic data are shown. A difference frame with a Laplacian statistical distribution, containing four regions that differ in their standard deviation, was constructed. Initial thresholds equivalent to MLE can be calculated since the σ values are known. In Fig. 7(a) the four region types are shown, where ST denotes 'stationary texture' ($\sigma = 13$), MT the 'moving texture' ($\sigma = 50$), SS 'stationary smooth' ($\sigma = 2$) and MS denotes 'moving smooth' ($\sigma = 11$). Fig. 7(b) shows the results of a simple threshold-based segmentation (with a threshold of 15), where a black dot indicates a moving-pixel decision. The poor result is due to the relatively high standard deviation of the stationary textured region. Fig. 7(c) shows the results obtained according to the method suggested in [9], also with a threshold of 15. It can be seen that the algorithm fails to recognize the moving smooth region and in addition, many pixels which belong to the stationary textured region are classified as moving. The effect of scanning along rows is also noticed. Fig. 7(d) shows the results obtained by the method suggested in [20] with a threshold of 15. There are almost no errors in the stationary textured region but again, the smooth moving region is not detected. Fig. 7(e) shows the results obtained by the proposed algorithm with a single resolution level. The initial thresholds used are 3 for smooth regions and 17 for textured regions. As can be seen, the algorithm separates almost exactly the moving (MT, MS) regions from the stationary (SS, ST) regions and it was found that varying the initial thresholds (used to determine X_0) had very little effect on the results. In Fig. 7(f) the results obtained by the proposed algorithm with two resolution levels are shown. The initial thresholds used, are as for a single resolution level. As can be seen, the segmentation is even better than the one in Fig. 7(e). The advantages of using two resolution levels, from the point of view of the cost-function value and the computational cost, are shown in Table 1. In both cases the algorithm was initialized with the same *non-optimal* thresholds (3 for smooth region and 17 for textured region). The computational cost is calculated as the 'average number of visits per pixel' as was done in [7].

In Fig. 7(g) the results obtained by the proposed algorithm with two resolution levels and *without* the texture/smooth segmentation are shown. It is clear that the moving smooth (MS) region is not detected, thus proving the usefulness of the TSS in the change detector. It can also be seen that even without the TSS, the results of the proposed change detector are better than those obtained by the algorithm suggested in [20] (shown in Fig. 7(d)). This is due to a better optimization process in the proposed change detector.

6.1.2. Simulation results for real data

In Fig. 8 the results obtained for real data, in comparing the proposed change-detection algorithm with other known change detectors, are presented. The source frames are frames 10 and 11 from the ISO test sequence 'table-tennis'. The poor segmentation in Fig. 8(a), obtained by the threshold-based method using a threshold of 8, is the result of the coarse textured background in the scene. Again, black indicates moving pixels and white indicates stationary pixels. In Fig. 8(b) the results obtained by the method suggested in [9] are shown. It can be seen that quite a large portion of the stationary background is labeled as moving and moving smooth regions on the sleeve are not detected. Again, the effect of scanning along lines in this model is noticed. In Fig. 8(c) the results obtained according to the method proposed in [20] are shown. The segmentation is smoother and the effect of lines is less noticeable in comparison with Fig. 8(b), but still a significant part of the background is labeled as moving. In Fig. 8(d) the results

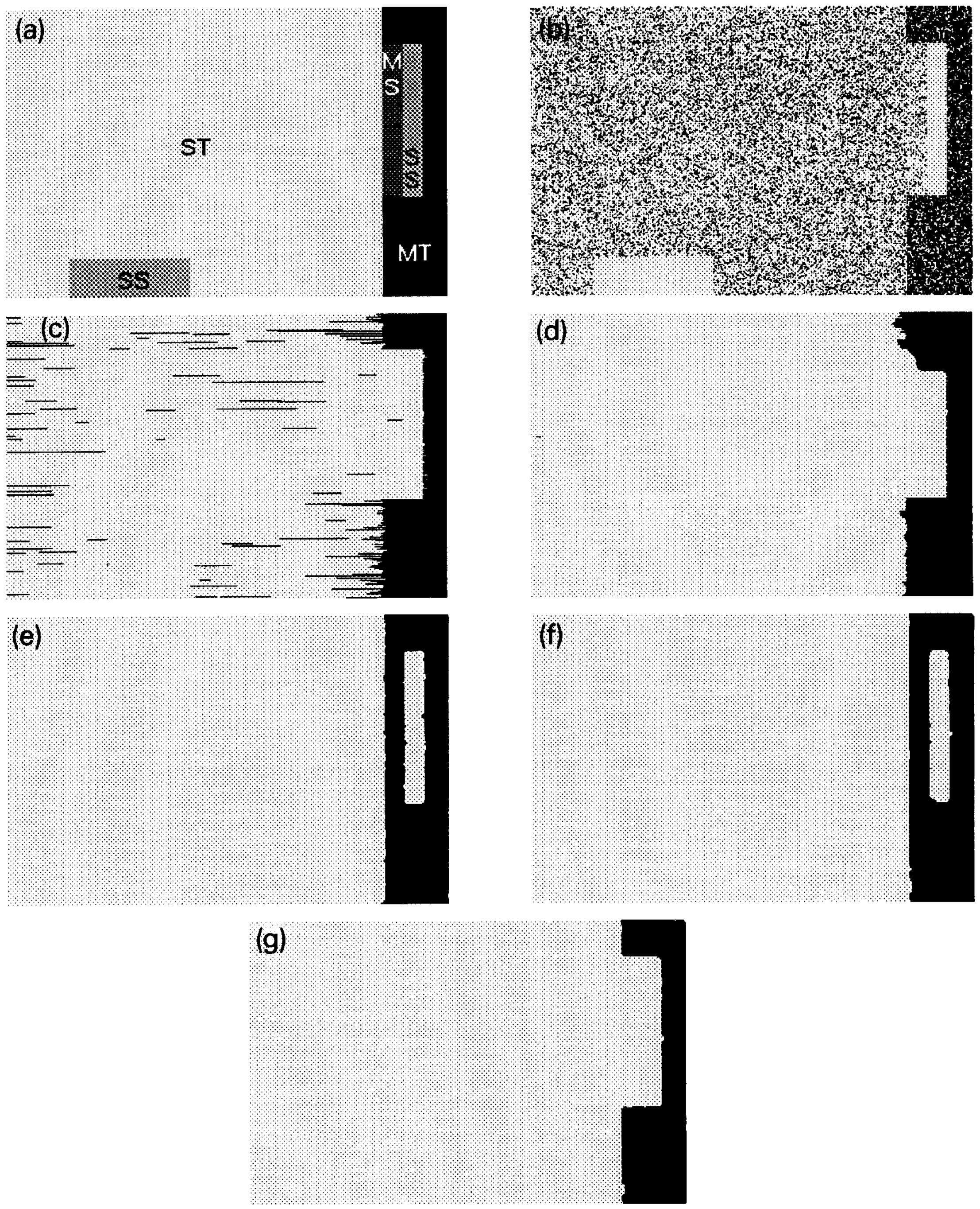

Fig. 7. (a) Different region types in the synthetic difference frame. ST indicates 'stationary texture', SS 'stationary smooth', MT 'moving texture' and MS 'moving smooth'. (b) Threshold-based segmentation results. (c) Results obtained by the method suggested in [9]. (d) Results obtained by the method suggested in [20]. (e) Results of the proposed algorithm with one resolution level. (f) Results of the proposed algorithm with two resolution levels. (g) Results of the proposed algorithm with two resolution levels but without the TSS.

Table 1
A comparison between minimization results and computational cost for one and two resolution levels

Number of resolution levels	Cost-function value $\times 10^{-2}$	Number of visits per pixel
1	3057	10.94
2	3042	5.78

of the proposed change detector with one resolution level are shown (with initial thresholds of 6 for smooth regions and 8 for textured regions). Better results are obtained with two resolution levels as can be seen in Fig. 8(e) (initial thresholds are 20 for smooth regions and 40 for textured regions). Since the proposed algorithm is less dependent on the initial thresholds for the initial segmentation than other algorithms, fixed thresholds values can be used for sequences of images with different characteristics. This was shown to be true when the change detector was tested on other sequences such as 'beach and flower' and 'flower-garden'.

6.2. Coding results for scenes without global motion

For testing the proposed block classification unit within an image sequence coding scheme, we constructed a coder which is based on the RM8 coder [1, 2]. The main advantage of incorporating the block classification unit into the coder is its ability to avoid the coding of stationary textured blocks. Such blocks are coded by the conventional RM8 coder because of the high variance in the corresponding difference block. The coder was used in coding the luminance part of the first 24 frames in the 'table-tennis' sequence, which contain *local* motion only on a coarse textured background. The value of T_B was chosen to be 1.3.

The classification results for frame No. 11 in the 'table-tennis' sequence are shown in Fig. 9(b). The corresponding original frame is shown in Fig. 9(a) for reference. White regions in Fig. 9(b) indicate blocks that are copied from the previous reconstructed frame, gray indicates blocks that are coded as a result of the texture-matching test and edge detection, and black indicates blocks that are coded as a result of the change detection. The small white bars in some of the blocks indicate the existence of a non-zero motion vector for the block (the length of the bar corresponds to the size of the motion vector and its orientation is proportional to the direction of the motion vector). As can be seen, blocks which belong to moving regions (like the arm, hand, bat and ball), along with blocks which contain edges are coded. Blocks that belong to the textured stationary background (which covers a major part of the frame) are copied, even when the corresponding difference block contains relatively high values. Some of the background blocks are characterized by non-zero motion vectors, as a result of stray motion. The copying of such blocks shows the ability of the proposed block classification unit to overcome the stray-motion problem.

Results obtained with the proposed coder in comparison with an RM8-type coder are summarized in Table 2, where q_s is the fixed quantization step used in comparing the coders.

As can be seen, a large reduction in the bit-rate is obtained using the proposed coder. The lower PSNR obtained by the proposed coder, as compared with the RM8-type coder with the same quantization step, is the result of copying those blocks which are classified as *textured background* blocks, even if they have a high prediction error. The difference in the PSNR is smaller when a larger quantization step is used in both coders, because the error introduced by quantization is increased (especially in the RM8-type coder which codes many more blocks than the proposed coder). The perceived quality of the reconstructed sequence (displayed in real time), when the *same quantization step* is used in the RM8-type coder and the proposed coder, was judged similar, in spite of the lower rate of the proposed coder. The quality of the reconstructed sequence was judged better than RM8 at the *same rate*. This can be explained by the lower quantization step used in the proposed coder.

The bit-rate as a function of the frame number is shown in Fig. 10. The results for a fixed quantization step of $q_s = 8$ are shown in Fig. 10(a). As can be seen, the fluctuations in the bit-rate and the average bit-rate of the proposed coder (dashed line) are much smaller than in the RM8-type coder (solid line). Similar behavior can be seen in Fig. 10(b) for

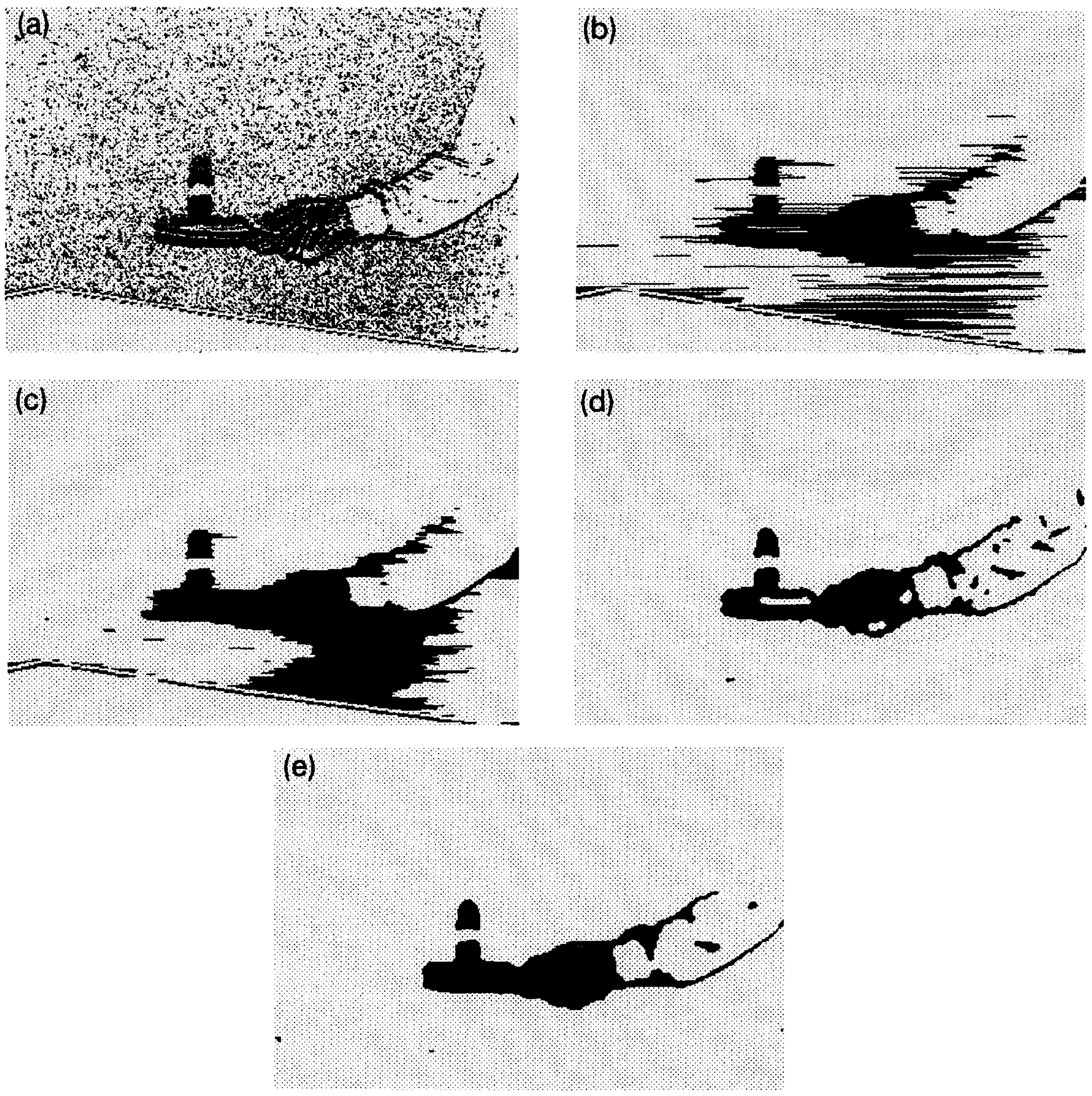

Fig. 8. Change-detection results for frames 10–11 in 'table-tennis'. (a) Threshold-based segmentation results. (b) Results obtained by the method suggested in [9]. (c) Results obtained by the method suggested in [20]. (d) Results obtained by the proposed algorithm with one resolution level. (e) Results obtained by the proposed algorithm with two resolution levels.

a fixed quantization step of $q_s = 16$. Note that the first frame is coded at the same rate in both coders.

6.3. Coding results for scenes with global motion

As mentioned in Section 5, incorporating a GMC unit enables the proposed coder to operate on scenes having global motion. We tested the coder on 2-D scenes with global motion. The tested sequence was 'table-tennis' (frames 0–100). Frames 24–88 of this sequence include a zoom-out operation, and there is a scene-cut on frame 89. Results obtained with the proposed coder in comparison with an RM8-type coder (with and without GMC) are summarized in Table 3, where q_s is the fixed quantization step used in the coders.

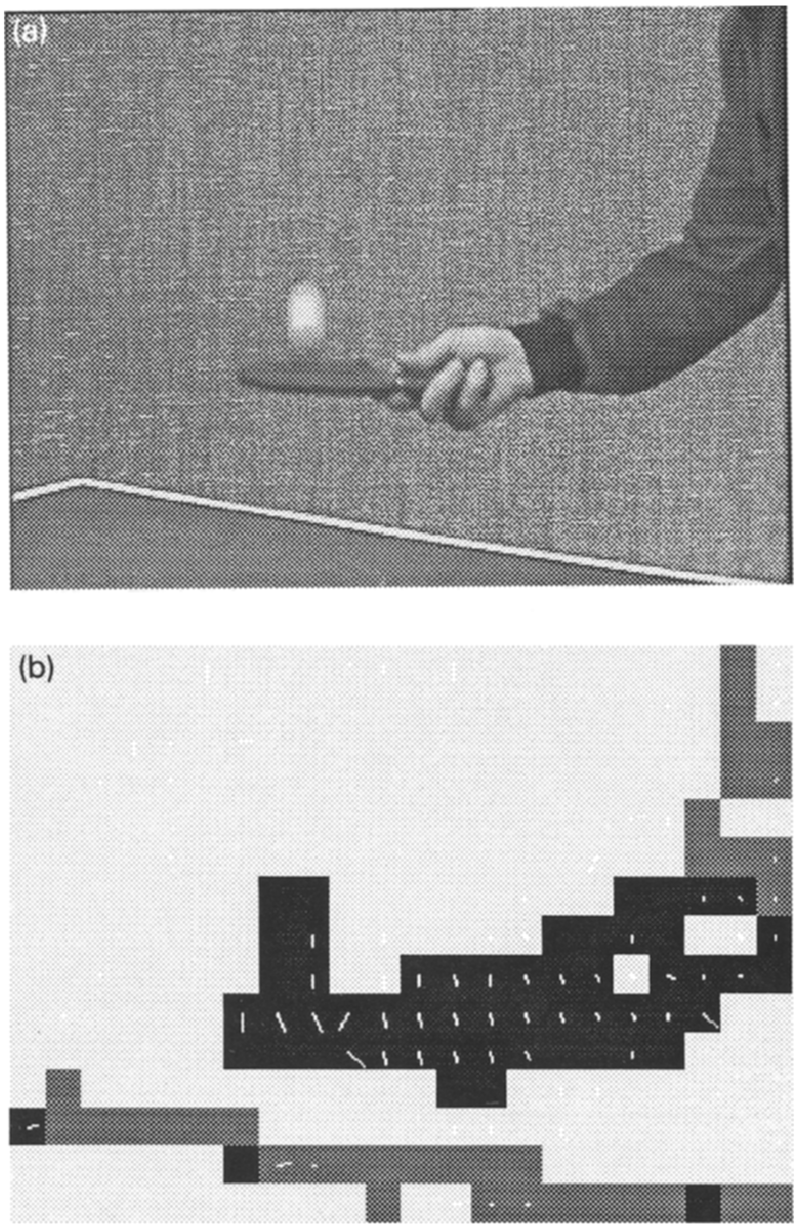

Fig. 9. (a) Original frame No. 11 from 'table-tennis'. (b) Block classification results where white indicates blocks that are copied, gray indicates blocks that are coded as a result of the texture-matching test and edge detection, black indicates blocks that are coded as a result of the change detection and the white bars indicate the existence of motion vectors (the length and orientation of the bars correspond to the size and direction of the motion vectors, respectively.

As can be seen in Table 3, the proposed coder (with an added GMC unit) enables a substantial reduction in the bit-rate for scenes having global motion as well. The resulting PSNR is closer to the value obtained wih the RM8-type coder than for scenes without global motion. This is because the zoom-out process causes the texture to become smoother, so that the undesirable smoothing effect

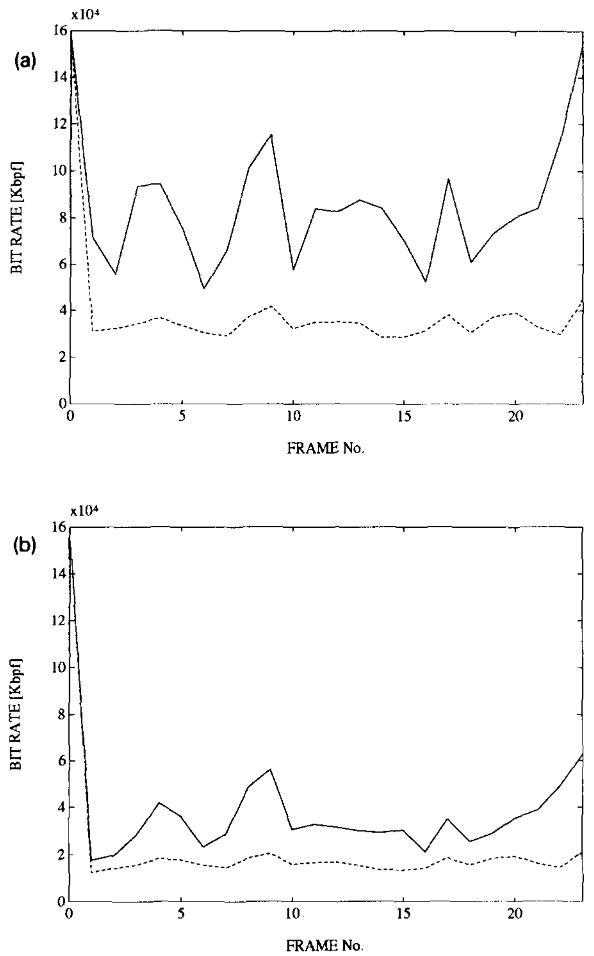

Fig. 10. Bit-rate as a function of the frame number (frames 0–23 in the 'table-tennis' sequence) for RM8-type coder (solid line) and the proposed coder (dashed line) for (a) fixed quantization step of 8; (b) fixed quantization step of 16.

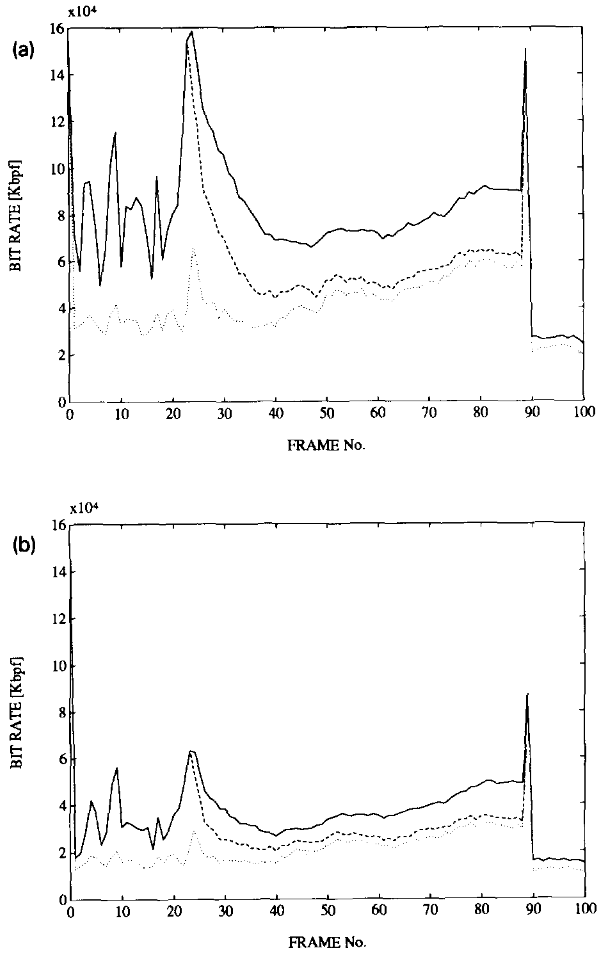

Fig. 11. Bit-rate as a function of the frame number (frames 0–100 in the 'table-tennis' sequence, zoom-out in frames 24–88) for RM8-type coder without GMC (solid line), RM8-type coder with GMC (dashed line) and the proposed coder (dotted line) for (a) fixed quantization step of 8; (b) fixed quantization step of 16.

Table 2
Bit-rate and PSNR values for an RM8-type coder, in comparison with the proposed coder, for a scene *without* global motion ('table-tennis', frames 0–23)

Coder	q_s	Average rate (Kbps)	PSNR (dB)	% Savings in bit-rate
RM8-type	8	2485	35.12	–
Proposed coder	8	1023	28.38	58.8%
RM8-type	16	1020	30.35	–
Proposed coder	16	489	27.80	52.1%

caused by the interpolation process used in the GMC unit [4] is reduced, resulting in a better compensation by the GMC unit. In addition, when the textured background becomes smoother, the role of the proposed change detector in reducing the number of coded blocks is diminished.

The last statement is demonstrated in Fig. 11. In Fig. 11(a), the bit-rate as a function of the frame number for a fixed quantization step of $q_s = 8$ is shown. As can be seen, the effectiveness of the proposed coder is more emphasized in the first 24 frames where no global motion is present. When

Table 3
Bit-rate and PSNR values for an RM8-type coder, in comparison with the proposed coder, for a sequence which contains global motion ('table-tennis', frames 0–100)

Coder	q_s	Average rate (Kbps)	PSNR (dB)	% Savings in bit-rate
RM8	8	2331	36.09	–
RM8 + GMC	8	1839	36.44	21.1%
Proposed coder + GMC	8	1251	33.86	46.3%
RM8	16	1059	32.13	–
RM8 + GMC	16	861	32.57	18.7%
Proposed coder + GMC	16	630	31.67	40.5%

global motion exists, the proposed coder improves the results as compared with the RM8-type coder (to which a GMC unit is added as well), but more modestly. Fig. 11(b) shows the same comparison but for a fixed quantization step of $q_s = 16$.

7. Summary and conclusions

An improved image sequence coder is proposed in this paper. The coder contains a block classification unit which enables a reliable detection of all stationary blocks including coarse textured background blocks. These blocks, which usually feature high variance in the corresponding difference block, can then be *copied* from the previous reconstructed frame, instead of being coded (as would be the case in the RM8-type coders). The unit consists of a change detector, which is based on statistical models for the difference frame (a Laplacian PDF) and for the moving/stationary segmentation field (Gibbs distribution function) and on a MAP-based cost function. An iterative sub-optimal optimization procedure (ICM) is used to obtain the final moving/stationary segmentation. The additional segmentation of each image into smooth/textured regions, and the directed use of several initial moving/stationary segmentation fields in the optimization procedure, enables the adaptation of the change detector to image characteristics and the reduction of its sensitivity to the initial segmentation. Performing the minimization procedure at several resolution levels enables further improvement in the segmentation results while lowering the computational cost. After the changed blocks are detected by the change detector, a texture-matching test and the detection of edges in all the blocks which were labeled as stationary are used to verify the validity of copying the stationary textured blocks. Incorporation of the proposed block classification unit in an RM8-type image-sequence coder results in a substantial reduction in the bit-rate of the coder for scenes with coarse textured background, while maintaining the perceived quality of the reconstructed sequence. Incorporating an adequate global motion compensation (GMC) unit in addition to the block classification unit enables the application of the proposed coder for coding also scenes which contain global motion. In this case, using a special reference frame, which reflects the accumulation of global motion parameters estimation errors (as described in Section 5) is necessary.

Acknowledgments

This work was supported by the Samuel Neaman Institute for Advanced Studies in Science and Technology of the Technion.

We thank the anonymous reviewers for their helpful comments which helped to improve the paper.

References

[1] "Description of Reference Model 8 (RM8)", Document 525, CCITT SGXV, Working Party XV/4, Specialist Group on Coding for Visual Telephony 1989.

[2] "Draft revision of recommendation H.261: Video codec for audiovisual services at $p \times 64$ kbit/s", *Signal Processing: Image Communication*, Vol. 2, No. 2, August 1990, pp. 221–239.

[3] H. Akaike, "A new look at the statistical model identification", *IEEE Trans. Automat. Control*, Vol. AC-19, No. 6, December 1974, pp. 716–723.

[4] A. Amitay, Global motion estimation in 3-D scenes for coding application, M.Sc. Thesis, Technion I.I.T, EE. Dept., Haifa, Israel, March 1993 (In Hebrew).

[5] A. Amitay and D. Malah, "Global motion estimation in image-sequences of 3-D scenes for coding applications", *Proc. Picture Coding Symposium – PCS'93*, Paper No. 10.1, Lausanne, Switzerland, March 1993.

[6] J. Besag, "On the statistical analysis of dirty pictures", *J. Roy. Statist. Soc. B*, 1986, pp. 259–302.

[7] C. Bouman and B. Liu, "Multiple resolution segmentation of textured images", *IEEE Trans. Pattern Anal. Machine Intell.*, Vol. PAMI-13, No. 2, 1991, pp. 99–113.

[8] P.J. Burt and E.H. Adelson, "The Laplacian pyramid as a compact image code", *IEEE Trans. Comm.*, Vol. COM-31, No. 4, April 1983, pp. 532–540.

[9] C. Cafforio and F. Rocca, "Methods for measuring small displacements of television images", *IEEE Trans. Inform. Theory*, Vol. IT-22, No. 5, 1976, pp. 573–579.

[10] N. Diehl, "Object-oriented motion estimation and segmentation in image sequences", *Signal Processing: Image Communication*, Vol. 3, No. 1, February 1991, pp. 23–56.

[11] J. Driessen, J. Biemond and D.E. Boekee, "A pel recursive segmentation and estimation algorithm for motion compensated image sequence coding", *Proc. Internat. Conf. Acoust. Speech Signal Process. 1989*, pp. 1901–1904.

[12] Z. Eisips and D. Malah, "Global motion estimation for image sequence coding applications", *Proc. 17th Convention of Electrical and Electronical Engineering in Israel*, May 1991, pp. 186–189.

[13] Z. Fan, "Textured image segmentation – An adaptive edge detection approach", *Proc. 22nd Asilomar Conf.*, Asilomar, CA, 1988, pp. 300–304.

[14] S. Geman and D. Geman, "Stochastic relaxation, Gibbs distribution, and the Bayesian restoration of images", *IEEE Trans. Pattern Anal. Machine Intell.*, Vol. PAMI-6, No. 6, 1984, pp. 721–741.

[15] C. Jones and K. Sauer, "A Bayesian approach to segmentation of temporal dynamic data in video data", *SPIE Visual Comm.*, Vol. 1605, 1991, pp. 522–532.

[16] K.P. Karmann, A.V. Brandt and R. Gerl, "Moving object segmentation based on adaptive reference images", in: *Signal Processing, Vol. V: Theories and Applications*, 1990, pp. 951–954.

[17] R.L. Kashyap, "Inconsistency of the AIC rule for estimating the order of autoregressive models", *IEEE Trans. Automat. Control*, Vol. AC-25, No. 5, October 1980, pp. 996–998.

[18] P. Lalande and P. Bouthemy, "A statistical approach to the detection and tracking of moving objects in an image sequence", in: *Signal Processing, Vol. V: Theories and Applications*, 1990, pp. 947–950.

[19] J.W. Modestino and J. Zhang, "A model-fitting approach to cluster validation with application to stochastic model-based image segmentation", *SPIE Visual Comm. Visual Processing IV*, Vol. 1199, 1989, pp. 1381–1392.

[20] L. Mori, F. Rocca and S. Tubaro, "Motion compensated interpolation using foreground/background segmentation", *Digital Signal Processing*, 1991, pp. 379–384.

[21] Y. Ozturk and H. Abut, "Application of the Itakura–Saito type spectral distortion measure to image analysis and classification", *Proc. 24th Asilomar Conf.*, Asilomar, CA, 1990, pp. 341–345.

[22] T.V. Papathomas and D. Malah, "Experimentally obtained thresholds for a conditional-replenishment image-sequence coder", *J. Visual Comm. Image Representation*, Vol. 4, No. 1, March 1993, pp. 79–91.

[23] Z. Sivan, Motion detection and texture analysis for image sequence coding, M.Sc. Thesis, Technion I.I.T, EE Dept., Haifa, Israel, March 1993 (In Hebrew).

[24] Z. Sivan and D. Malah, "Change detection for image sequence coding", *Proc. Picture Coding Symposium –PCS'93*, Paper No. 14.1, Lausanne, Switzerland, March 1993.