Object Based Video Coding by Global-to-Local Motion Segmentation

Ahsan Shamim and John A. Robinson

***Abstract*—In this paper, we describe an object-based video compression scheme based on the derivation and efficient coding of motion boundaries. First, we recursively identify a small number of *global movement classes*, each represented by two or more motion parameters. Second, we assign regions of a spatial segmentation to movement classes. Third, we merge segments using various similarity heuristics, while adding movement classes for small objects, if necessary. Finally, the boundaries of motion are coded using an efficient asymmetric binary tree (ABT) coding scheme. Experimental results on standard test sequences qualitatively show that the proposed algorithm gives good segmentation, and that it is suitable for very-low-data-rate object-based coding.**

***Index Terms*—Motion estimation, motion segmentation, video coding.**

I. Introduction

VIDEO compression schemes exploit temporal redundancy by motion estimation, motion-compensated prediction, and residue coding. Standards such as H.263, MPEG 1 and 2 use block-based motion compensation and a discrete cosine transform (DCT) for block-based coding of prediction residues [1]. The drawback of these algorithms is that real-world moving objects, in general, do not align with the block boundaries. Near the object boundaries, block-oriented algorithms derive erroneous motion vectors, leading to discontinuity in the motion-vector fields. Block-based residue coding then produces blocking artifacts in the reconstituted picture. The human visual system is very sensitive to such artifacts in an image.

Object-based video coding identifies moving objects in an image sequence, then codes their boundaries and interiors separately. It achieves more efficient compression than block-oriented schemes, without blocking artifacts. MPEG 4 and MPEG 7 support object- and content-based techniques. Examples of approaches to detecting, segmenting, and coding moving objects in a video sequence include [2]–[8]. In this paper, we propose a new scheme that is unusual in developing motion estimates in a hierarchical, global-to-local, fashion. We also explain how our asymmetric binary tree (ABT) coding scheme [9] is applied to the results of segmentation to code object boundaries efficiently. Fig. 1 shows a block diagram of the system as a whole.

Manuscript received August 3, 2001; revised August 2, 2002. This paper was recommended by Associate Editor K. Aizawa.

A. Shamim was with the Faculty of Engineering and Applied Science, Memorial University of Newfoundland, St. John's, NF A1B3X5, Canada. He is now with the Nokia Mobile Phones (R&D), Salo 24101, Finland (e-mail: Ahsan.Shamim@nokia.com).

J. A. Robinson was with the Faculty of Engineering and Applied Science, Memorial University of Newfoundland, St. John's, Newfoundland A1B3X5, Canada. He is now with the Department of Electronics, University of York, Heslington, York YO10 5DD, U.K. (e-mail: Jar11@ohm.york.ac.uk).

This work was conducted when both authors were at Faculty of Engineering and Applied Science, Memorial University of Newfoundland, St. John's, Newfoundland A1B3X5, Canada.

Digital Object Identifier 10.1109/TCSVT.2002.806811

II. Region-Based Motion Estimation

A. Determining Movement Classes

For simplicity, we consider only the coding of a single current frame, using a single previous frame for motion estimation and prediction. Extension to bidirectional and multiframe coding is possible but not developed here. We first consider the whole image as a single moving object and, by gradient descent, find a best-estimate translational motion estimate. This generally represents background or camera motion, and parts of the image that have this motion are in the first *movement class*. The matching errors that result from displacing the previous frame by this motion and subtracting the current frame are thresholded. Following a morphological operation to remove noise, the pixels with matching error greater than the threshold are judged as outside the first movement class, and are considered as a (noncontiguous) region for a second round of gradient-descent motion estimation. The best-estimate motion parameters for this new region define the second "movement class" and, again, the difference image is thresholded and the above-threshold points used to form a new region. The process repeats until most of the picture is subthreshold for one of the motion vectors. In general, this technique finds the motion vectors of large contiguous regions before those of smaller regions. After a certain stage of motion vector calculation, only small unrelated objects remain. At this stage, gradient-descent techniques become unreliable. So, a full-search algorithm is applied if the cross-correlation coefficient between two test frames with remaining objects falls below some threshold (i.e., 0.1) at an estimated motion vector.

B. Threshold Calculation

The process of determining movement classes described in Section II-A requires repeated thresholding of matching errors. Thresholds are determined automatically, and are of two types: true motion and false motion. These types are identified and processed as follows. First, a translational motion estimate is found using all of the points not previously classified. Next, the cross-correlation profile between the reference frame and the displaced frame is calculated using pixels with matching errors greater than thresholds from 0 to 255. This profile has several curve segments that decrease exponentially. In general, points in each segment correspond to related objects. Those points having matching errors in the first segment are assumed to follow the calculated motion vector. We, therefore, calculate the threshold

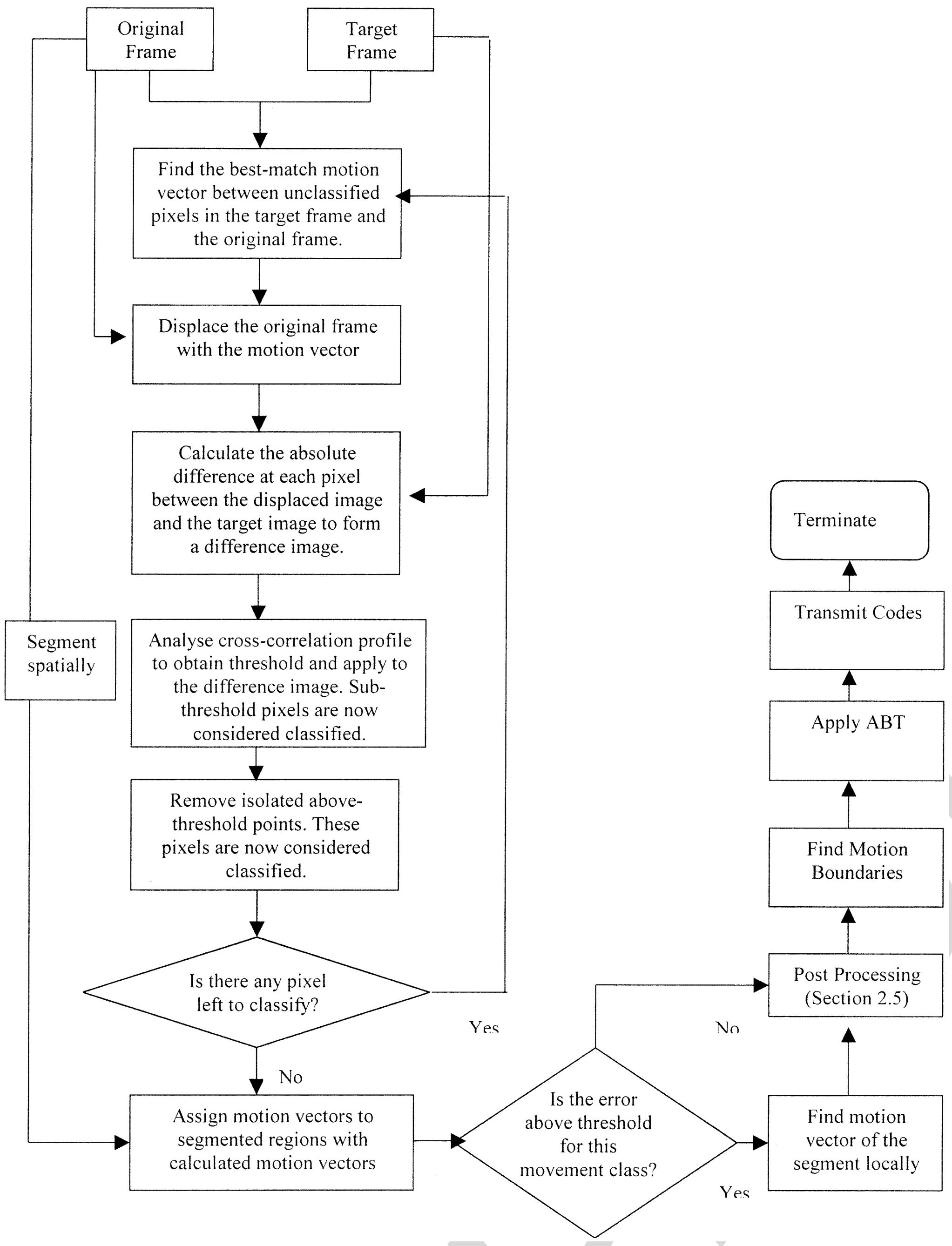

Original Frame
Target Frame
Find the best-match motion vector between unclassified pixels in the target frame and the original frame.
Displace the original frame with the motion vector
Calculate the absolute difference at each pixel between the displaced image and the target image to form a difference image.
Segment spatially
Analyse cross-correlation profile to obtain threshold and apply to the difference image. Sub-threshold pixels are now considered classified.
Remove isolated above-threshold points. These pixels are now considered classified.
Is there any pixel left to classify?
Yes
No
Assign motion vectors to segmented regions with calculated motion vectors
Is the error above threshold for this movement class?
No
Yes
Find motion vector of the segment locally
Post Processing (Section 2.5)
Find Motion Boundaries
Apply ABT
Transmit Codes
Terminate

Fig. 1. Block diagram of the method.

from the point of maximum slope between the first and second segments. Fig. 2(a) and (b) shows the threshold calculation technique of the two general types of profiles for true motion and false motion.

Fig. 2(a) shows the natural correlation profile for true motion vectors. The range from 0 to t_1 corresponds to the same correlation level. Then, the correlation coefficient decreases until the threshold reaches t_m. Near t_m, the profile becomes flat, as the effect of other objects being removed after a certain threshold becomes prominent. In order to avoid the removal of other objects moving with different motion vectors, the maximum slope of the profile between 0 and t_m is found, and the desired threshold t_2 is obtained as follows:

$$t_2 = t_1 + \frac{(\rho_{\max} - \rho_{\min})}{\delta_{\max}} \tag{2.1}$$

where $\delta_{\max}$ is the maximum gradient of the profile between 0 and t_m.

Fig. 2(b) shows the correlation profile for false motion vectors. Generally, this type of profile results from false responses of the remainder of objects whose motion vectors have already

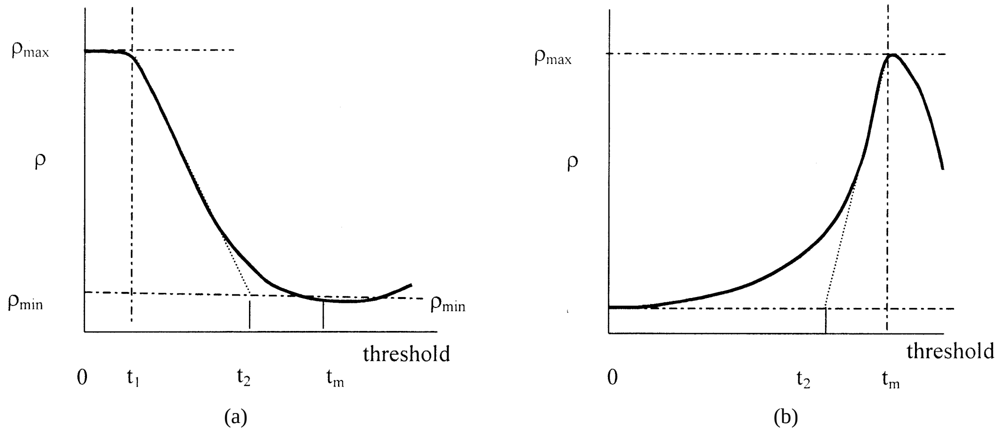


Fig. 2. Cross-correlation coefficient (ρ) versus threshold profile for (a) true motion vector and (b) false motion vector.

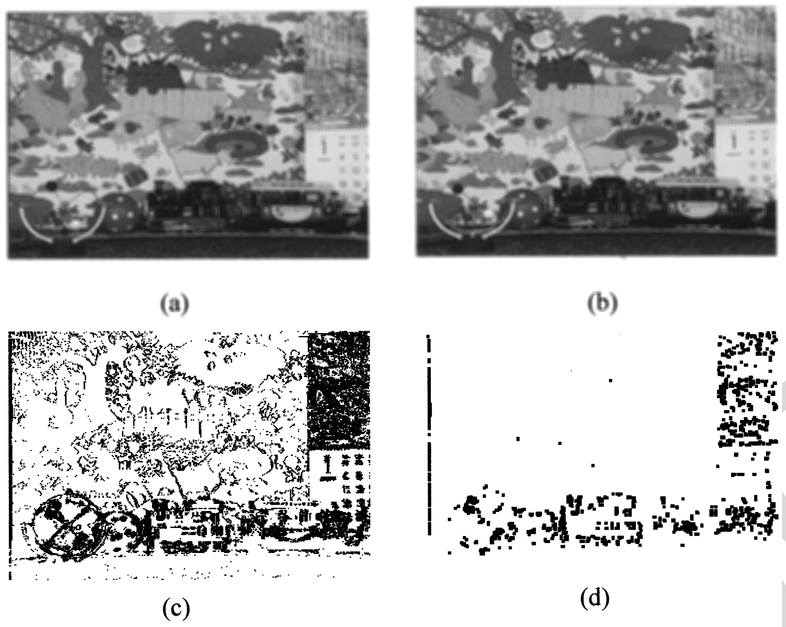


Fig. 3. (a) Frame 91 and (b) Frame 94 of the *mobile and calendar* sequence. (c) Thresholded image after the first stage of motion vector calculation. White points belong to the first movement class; black points are not yet classified. (d) Remaining unclassified points (in black) after "opening" operation on the thresholded image in (c). These points will now be used to find the motion vector for the second movement class.

been found. The motion vector for this type of profile is therefore not used to define a movement class. However, threshold t_2 is calculated as follows:

$$t_2 = t_m - \frac{(\rho_{\max} - \rho_{\min})}{\delta_{\max}}. \tag{2.2}$$

The points that yielded cross-correlation profile values below threshold t_2 are then eliminated from the next level of the motion vector search. That is, they are assumed to belong to one of the previously derived movement classes, and their removal ensures that they do not adversely affect the calculation of the next true motion vector. The process of calculating motion vectors continues until the profile becomes flat, implying that the remaining part of the picture will not contribute any more valid motion vectors.

C. *Morphological Operation*

In order to remove noise from one frame to another, *opening* is done on the thresholded difference image with a 3×3 square operator. The opening of set A by structuring element B, denoted $A \circ B$, is defined as [6] $A \circ B = (A\Theta B) \oplus B$, where $(A\Theta B) = \{x|(B)_x \subseteq A\}$ is the erosion of A by B, which corresponds to the set of all points x such that B, translated by x, is contained in A and $(A\Theta B) \oplus B$ is the dilation of $(A\Theta B)$ by B, which represents the set of all x displacements such that $\hat{B} = \{x|x = -b, \text{for } b \in B\}$ and $(A\Theta B)$ overlap by at least one nonzero element.

The main attribute of *opening* is that it smoothes the contours of an image, breaks narrow isthmuses, and eliminates thin protrusions. It is appropriate for our purpose of determining movement classes because practical objects in general have smooth surfaces. Fig. 3 shows the effect of an *opening* operation in estimating motions between two frames in the *mobile and calendar* sequence.

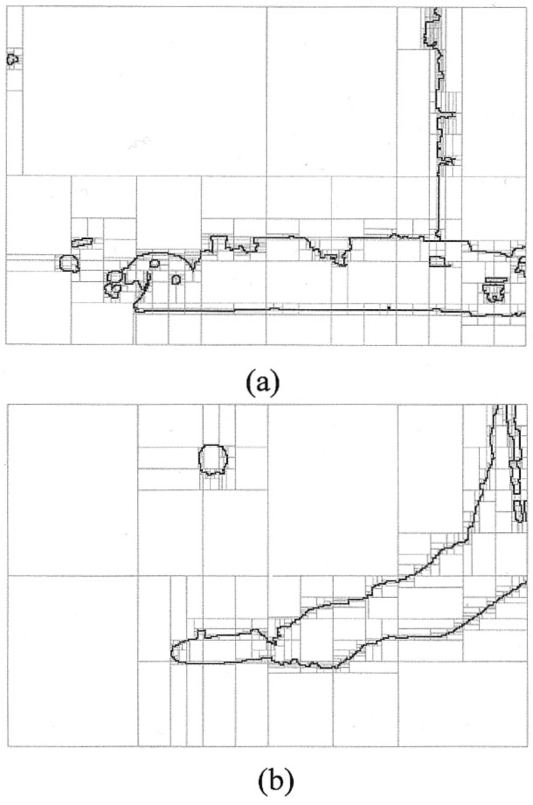


Fig. 4. Application of the ABT coding scheme on a motion boundary image of (a) the *mobile* and *calendar* sequence and (b) the *table tennis* sequence.

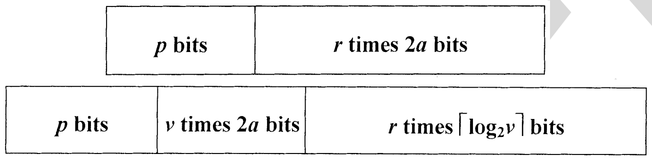


Fig. 5. (a) First strategy and (b) second strategy of header arrangement.

The handling of false motion vectors (Section II-B) and the morphological operation of opening (Section II-C) both remove unclassified pixels. These pixels are considered to belong to one of the previously found movement classes and, therefore, do not participate in the estimation of further movement classes. The actual classification of these pixels to particular movement classes does not happen until the segmentation and assignment of motion vectors (Section II-D).

D. Segmentation and Assignment of Motion Vectors

Areas of constant or slowly varying luminance in an image usually correspond to the surface of a rigid object. So, having obtained the movement classes, we spatially segment the reference frame using a region-growing algorithm, and then test each segment for each of the possible motion vectors. The motion vector giving the minimum matching error in each segment is considered to be applicable in that region, unless its matching error per pixel is greater than some threshold. The threshold is set at t_m (Fig. 2), the worst possible threshold to move with that motion vector, at the stage when that motion vector was found. When the threshold is exceeded, the algorithm performs a local search for the best-matching region. If this gives matching error better than that from the best movement class vector, a new movement class is added, and the region is assigned appropriately. This updating process replaces a few previously assigned motion vectors with new ones if there are a significant number of smaller moving objects. Usually, the number of motion vectors calculated at the first stage is unaffected.

TABLE I
CODEWORDS USED IN THE ABT CODING SCHEME

	Primary Block	*Advanced Block*
For (m×n) blocks,		
Activity in both blocks (Split Equally):	A →0	D →0
Activity only in the first block:	B →10	E →1
Activity only in the second block:	C →11	F →1
For (m×1) and (1×m) blocks,		
Activity in both blocks (Split Equally):	G →00	K →0
Activity only in the first block:	H →01	L →10
Activity only in the second block:	I →10	M →10
All_black block:	J →11	N →11
For (2×1) and (1×2) blocks,		
Activity in both blocks:	O →0	R →0
Activity only in the first block:	P →10	T→1
Activity only in the second block:	Q →11	T→1

E. Post-Processing Strategies

After the initial stage of motion vector assignment, we apply three steps of post processing. First, if there is an object that is one-pixel wide and has different motion vectors from its surrounding, we consider this segment to have resulted from luminance change along an edge. In this case, we merge the segment to the surrounding, which gives a minimum matching error. The process repeats until no more merging can be done. Second, we check smaller objects with their surrounding motion vector. If the matching error with its present vector is less than s times that of its surrounding most likely motion vector, then the segment is merged with the surrounding. The purpose of this step is to clean up the segmentation of rigid objects that are rotating, scaling, or shearing, and which therefore consist of adjacent areas with similar translational motion vectors. A value of 1.1 for s merges a large percentage of these cases over a broad range of picture activity, and consistently results in a quality/data-rate improvement. Third, we assume a minimum object size of 4×4 pixels. If the segment is smaller, it is assumed to move with its surrounding motion vector, giving minimum matching error. The block size used for morphological operation is 3×3, which removes isolated clusters of pixels less than 4×4 in size. But near the end of the processing, the pixels used to calculate the motion vector are sparsely

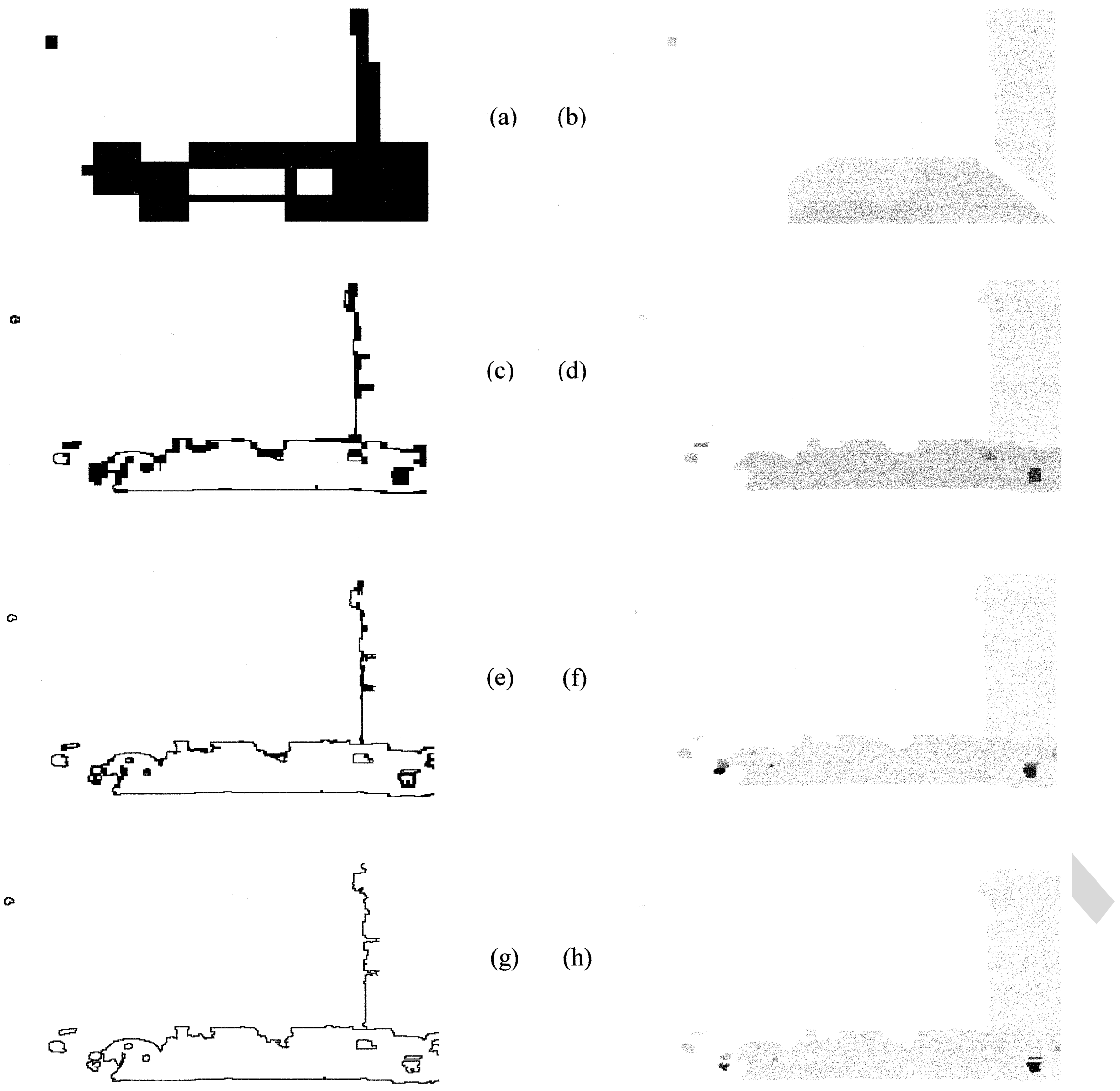


Fig. 6. (a), (c), (e), (g) Transmitted and decoded motion boundary image. (b), (d), (f), and (h) Recovered image after merging of black regions after processing stages 7, 12, 14, 17.

distributed, sometimes resulting in isolated clusters of pixels in size less than 4×4 being assigned a motion vector. This final post-processing stage cleans up these remaining small clusters. With these post-processing steps, the quality of the prediction image—the estimated frame constructed from the first frame of a sequence and the estimated motion vectors—does not degrade too much as compared to the original frame, but the motion segmentation is simplified, which facilitates efficient coding.

III. Coding

The coded (compressed) information consists of two main parts. First, the header information, which includes the total number of motion vectors, their values, and the number of segmented regions with different motion vectors. Second, the information describing the structure of differently moving segmented regions.

A. Motion-Information Coding

One of two strategies is followed to arrange the header information depending on the total number of motion vectors and total number of segmented regions with different motion vectors. In the first, bits representing the total number of segmented regions are transmitted then the measured motion vectors (i, j) of sequentially obtained segmented regions are transmitted. In the second strategy, bits representing the total number of vectors are transmitted, followed by the measured motion vectors (i, j), then the motion indices of sequentially obtained segmented regions.

Let us consider that p bits are transmitted to represent a maximum number of 2^p motion vectors or segmented regions. a bits are required to represent each component of the motion vector, and there are v total vectors and r total regions. Fig. 5 shows the difference between these two header arrangement strategies. Either of these two strategies is used, and 1 bit is transmitted initially to indicate the selection of the technique. For the performance comparison a is assumed to be equal to 6, which allows a

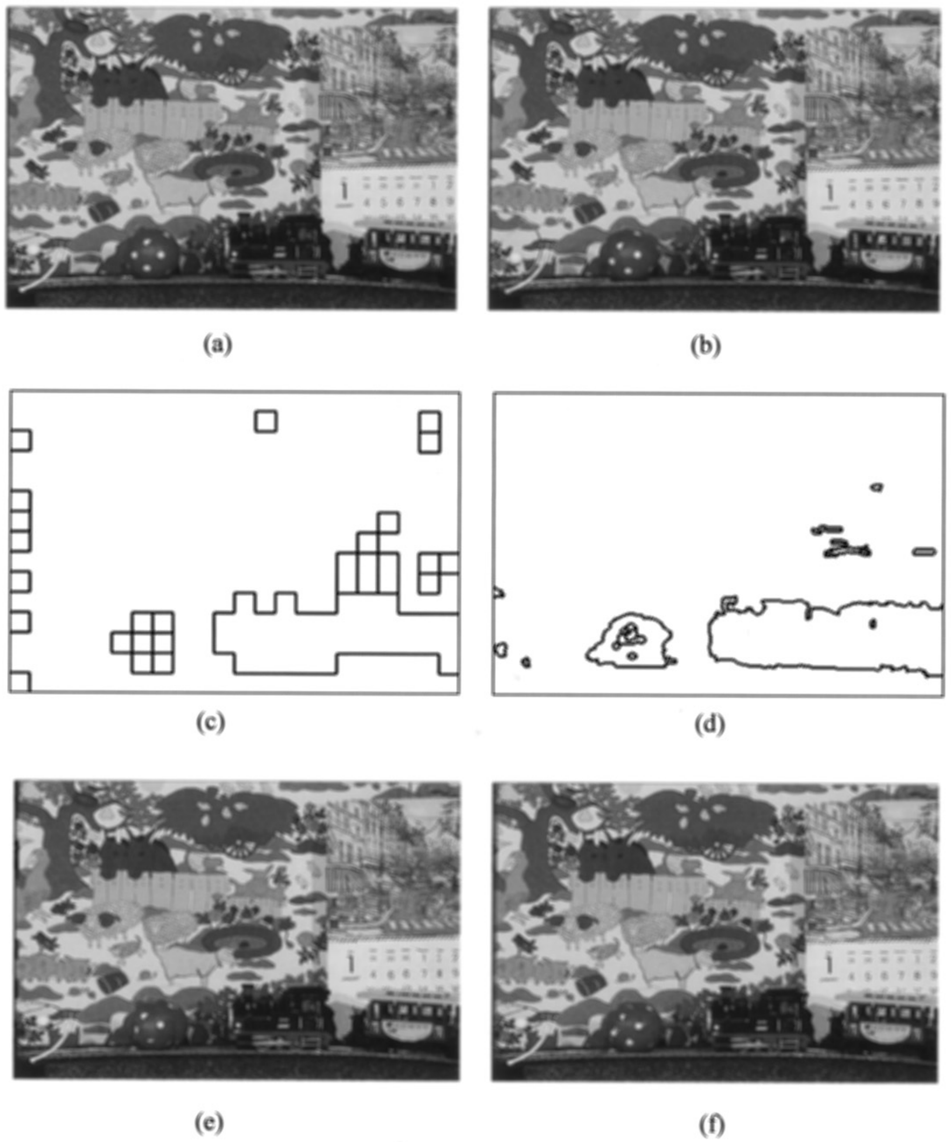


Fig. 7. (a) Frame 20 and (b) Frame 23 of the *mobile and calendar* sequence. (c) Motion vector boundary with block-matching technique. (d) Motion vector boundary with the proposed technique. (e) Predicted image with lock-matching technique [$\mathrm{PSNR} = 21.19$]. (f) Predicted image with the proposed technique [$\mathrm{PSNR} = 21.61$].

maximum codeable motion range of ± 32, greater than that typically used in conventional block-based coders.

B. Asymmetric Binary Tree Coding (ABT)

ABT codes the shapes of moving regions in a hierarchical way, similar to two-level coding schemes like quadtree and binary-tree coding [10],[11]. It shares with these earlier schemes the idea that rectangular blocks of the image are recursively subdivided until they contain all boundary or all interior pixels. In contrast to earlier methods, ABT adapts the cut position of a block. Moreover, no pixel is coded if its value is implicit in the higher stages of the hierarchy. Initially, all blocks are considered to be in the *primary stage*. ABT cuts a block in half if there are boundary pixels in both halves, and each half becomes a new *primary stage* block. Otherwise, it adjusts the cut position to maximize the size of the subblock without boundary pixels. The location of the cut is efficiently coded, together with a mark denoting which of the two subblocks has boundaries. When the subblock with boundaries is subsequently coded, it is in the *advanced stage*, because ABT exploits the constraint that boundary pixels must be adjacent to the most recent cut. The cut direction for an *advanced stage* block is the same as the previous cut; for a *primary stage* block, the direction is perpendicular to the previous. Codewords and a mapping to binary for the ABT scheme are shown in Table I. Here, the first column corresponds to *primary stage* blocks whereas the second column corresponds to *advanced stage* blocks. A detailed analysis of the scheme is given in [9]. Fig. 4 shows the application of ABT on two contour images; the thin lines show the division of the picture by ABT, illustrating how it iteratively splits rectangular blocks into two unequal subblocks.

Even with ABT, the structure information takes a large number of bits compared with the header information. Clearly

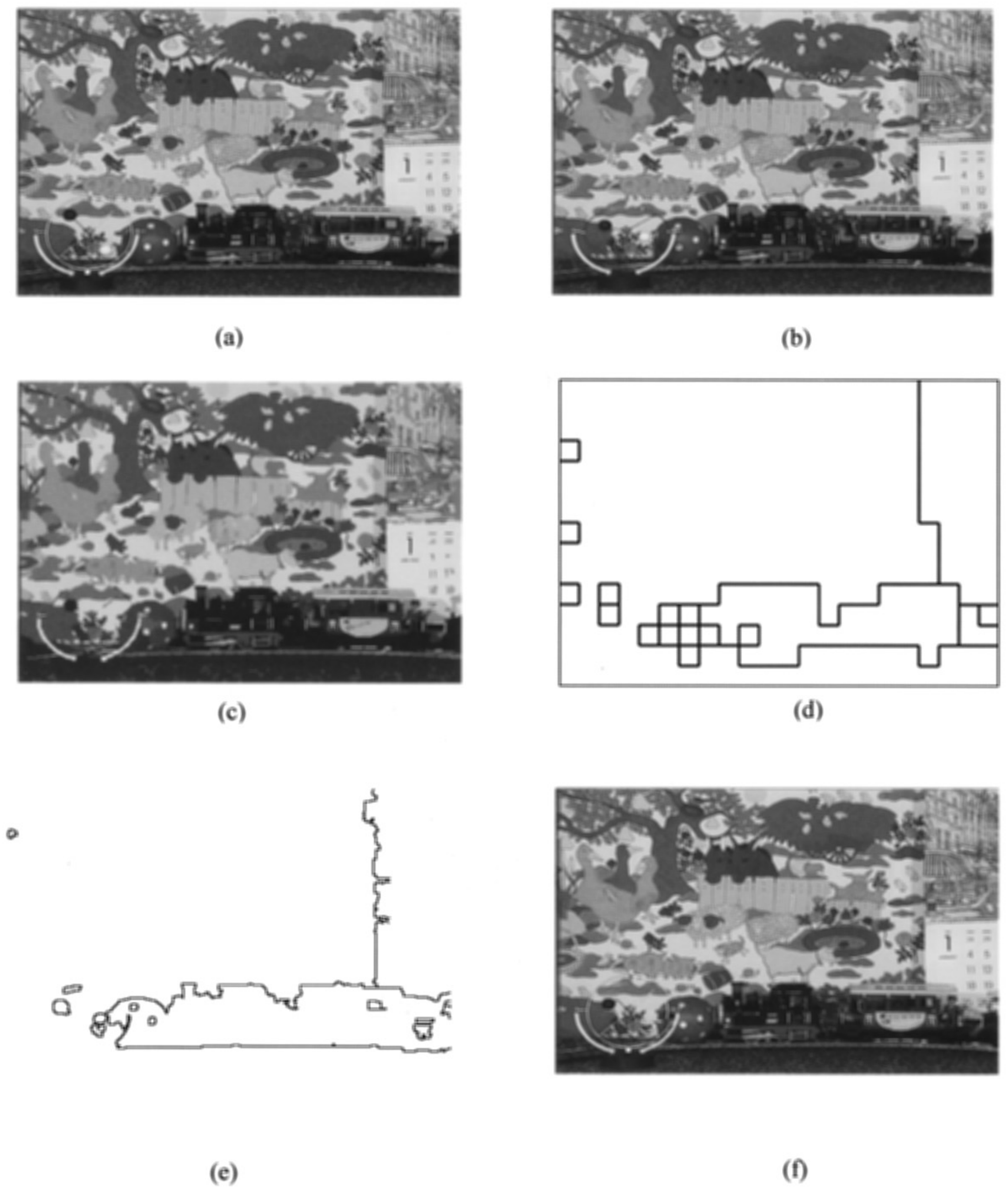


Fig. 8. (a) Frame 93 and (b) Frame 96 of the *mobile and calendar* sequence. (c) Segmented image of Frame 96 (1261 segments). (d) Motion vector boundary with block-matching technique. (e) Motion vector boundary with the proposed technique. (f) Predicted image with the proposed technique $[\mathrm{PSNR} = 22.47]$; the PSNR with block-matching technique is 21.44.

a multiframe coding scheme for moving objects could improve on what has already been achieved.

C. Variation of Reconstructed Image Quality With Variable Bit Rate

Because ABT is a hierarchical scheme, it allows an approximate version of the boundary information, and thus a block segmentation, to be derived from a partial encoding. This allows us to trade off the bits spent on the motion boundaries with the quality of the prediction image.

If coding stops before reaching the finest level, then there will be some blocks in an uncoded condition. These blocks need to be estimated from the coded blocks. When an uncoded block is at the intersection of different coded regions, these regions are grown across the block until they meet in boundaries inside. When the uncoded block is surrounded by a single coded region, the coded region is extended inwards in all directions to absorb half the uncoded area. The remaining uncoded area is defined as a single new region.

The transmitter needs to transmit a number of motion vectors equal to the sum of the number of coded regions and the number of isolated uncoded regions. During transmission, the encoder will transmit one motion vector for each coded region according to the scanning process. Then a motion vector for each isolated black region is transmitted, which is the most probable motion vector within the possible newly created region. All these motion vectors are used to form the header information. In this case,

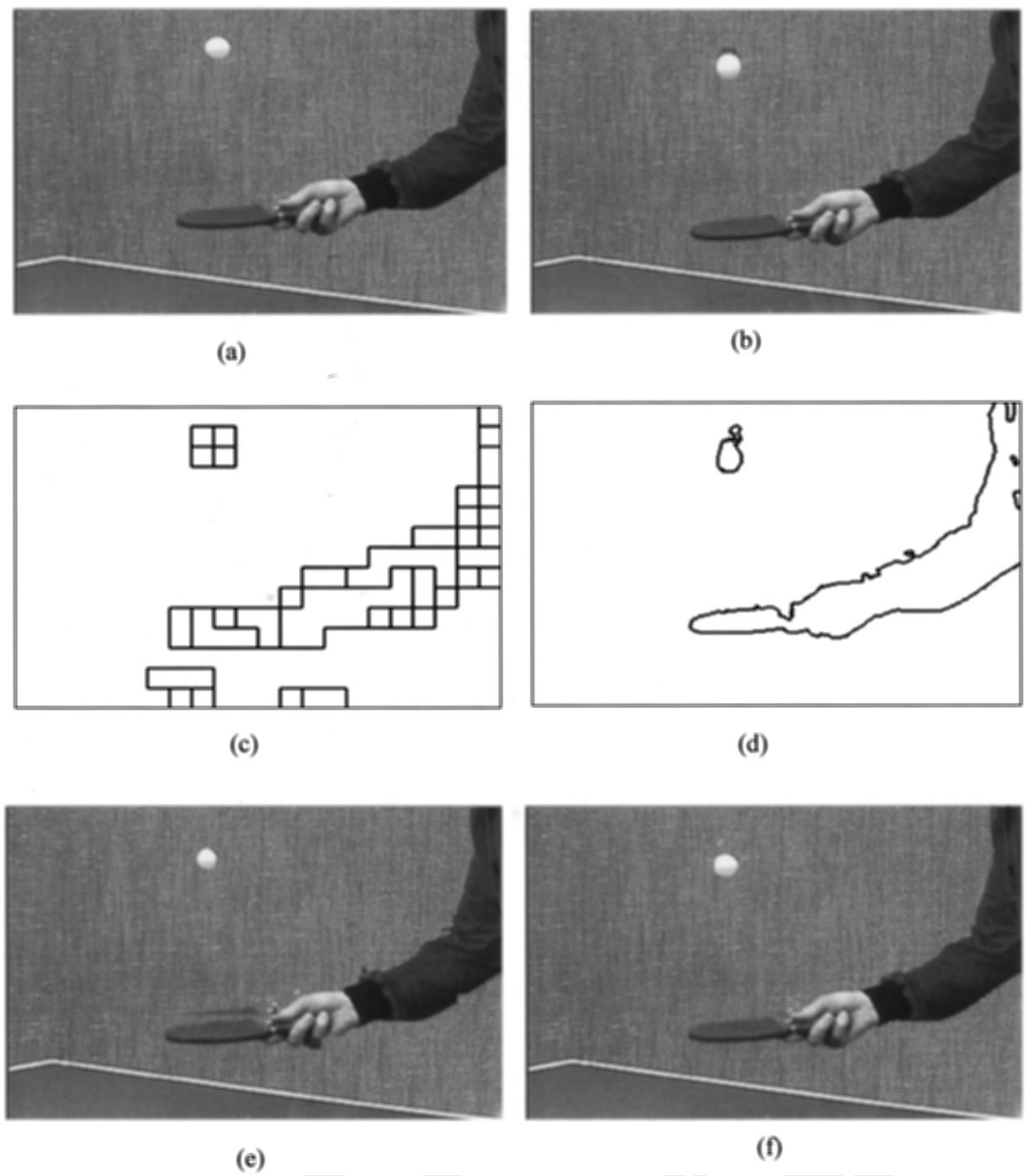


Fig. 9. (a) Frame 4 and (b) Frame 7 of the *table tennis* sequence. (c) Motion vector boundary with block-matching technique. (d) Motion vector boundary with the proposed technique. (e) Predicted image with block-matching technique $[\mathrm{PSNR} = 28.97]$. (f) Predicted image with the proposed technique $[\mathrm{PSNR} = 29.53]$.

the first strategy of arranging header messages is used (see Section III-A).

Fig. 6 shows the decoded region, and estimation of the uncoded region from the decoded region, of the motion boundary image between Frame 93 and Frame 96 of the *mobile and calendar* sequence. For example, Fig. 6(a) shows that after stage 7 of ABT, large areas (in black) are uncoded. Fig. 6(b) shows differently shaded regions corresponding to different moving objects based on the segmentation of Fig. 6(a); where Fig. 6(a) is white, the appropriate motion vector is represented by a particular shade of gray, and where Fig. 6(b) is black, the regions have been grown across the uncoded area as described above.

IV. Results and Discussion

Figs. 7–9 show motion vector boundaries and prediction pictures of a standard block-matching algorithm with (16×16) block size and of the proposed algorithm. These examples show that the proposed method yields a qualitatively good segmentation, except in small areas where noise on luminance edges causes incorrect extraction of small segments. By contrast, the shortcomings of block-oriented motion estimation motion boundaries are visible in all examples, particularly Fig. 9.

Table II shows the number of bits required to code the motion boundary information, the total number of motion vectors to be transmitted, the PSNR of the predicted image, and the bit rate assuming 30 frames per second, with respect to the ABT processing stage. Frames 93 and 96 of the *mobile and calendar* sequence are used for this example. We assume that 6 bits (maximum 64 regions) are required to transmit the information about the total number of motion vectors and 6 bits (± 32 pixel movement) are required to transmit each component of the motion vector.

TABLE II
VARIATION OF QUALITY OF THE PREDICTED IMAGE WITH BIT RATE USING TWO FRAMES OF THE *MOBILE AND CALENDAR* SEQUENCE

Stage No.	Transmitted bits for the boundary (A)	Total regions (B)	PSNR	Bit Rate (6+12×B+A) ×30 *bits/sec*
3	15	2	18.0919	1350
4	24	3	19.7208	1980
5	35	3	19.7739	2310
6	85	5	20.5075	4530
7	137	6	21.0653	6450
8	228	4	21.164	8460
9	366	7	21.4854	13680
10	598	7	21.5713	20640
11	961	9	21.8884	32250
12	1478	10	22.0345	48180
13	2100	14	22.0491	68220
14	2809	20	22.2082	91650
15	3612	23	22.366	116820
16	4204	19	22.3252	133140
17	4351	17	22.3158	136830

Fig. 10 shows the variation of PSNR with respect to the processing stage and bit rate. Some interesting analysis can be obtained from Fig. 10 and Table II.

- The total number of processing stages is 17. This is the maximum number of processing stages using any frame of size (240 × 352), the size of Frame 93 and Frame 96 of the *mobile and calendar* sequence. The reason for this is that the proposed coding technique uses a binary tree. In the worst case, there may be a block at the final stage to be coded that is obtained by equally splitting the corresponding previous-stage blocks until the final-stage subblock is reduced to a single active pixel. Hence, the total number of stages required to code this sub block is $\lceil \log_2 240 \rceil + \lceil \log_2 352 \rceil = 8 + 9 = 17$. In comparison, the maximum value of the number of processing stages in a quadtree is 7, because a quadtree divides a block both horizontally and vertically at the same processing stage.
- The processing stage in the proposed scheme required to get the predicted image of the same quality as that of block-matching technique deserves some attention. For an $(m_1 \times m_2)$ block size, if there is a break in motion within a block then the whole block of size $(m_1 \times m_2)$ will get a wrong motion vector. On the other hand, if the transmitter of the proposed technique stops transmission after a certain processing stage $\boldsymbol{p} = \boldsymbol{p_r} + \boldsymbol{p_c}$, then the worst possible uncoded block in the decoder will be of size $(2^{(mr-pr)} \times 2^{(mc-pc)})$, where p_r is the number of horizontal processing, p_c is the number of vertical processing, mr is equal to $\lceil \log_2(\text{number of rows of the image}) \rceil$, and mc is equal to $\lceil \log_2(\text{number of rows of the image}) \rceil$. This block is equivalent to getting a wrong motion vector in the conventional block-matching technique. The number of processing stages required in the proposed technique to get a predicted image of the same quality using block matching with block size $(m_1 \times m_2)$ can be theoretically calculated, and is approximately equal to the sum of $(mr - \log_2 m_1)$ and $(mc - \log_2 m_2)$. When $m_1 = 16$ and $m_2 = 16$, then $m_r = 8, m_c = 9, p_r = 4, p_c = 5$ and $p = 9$.

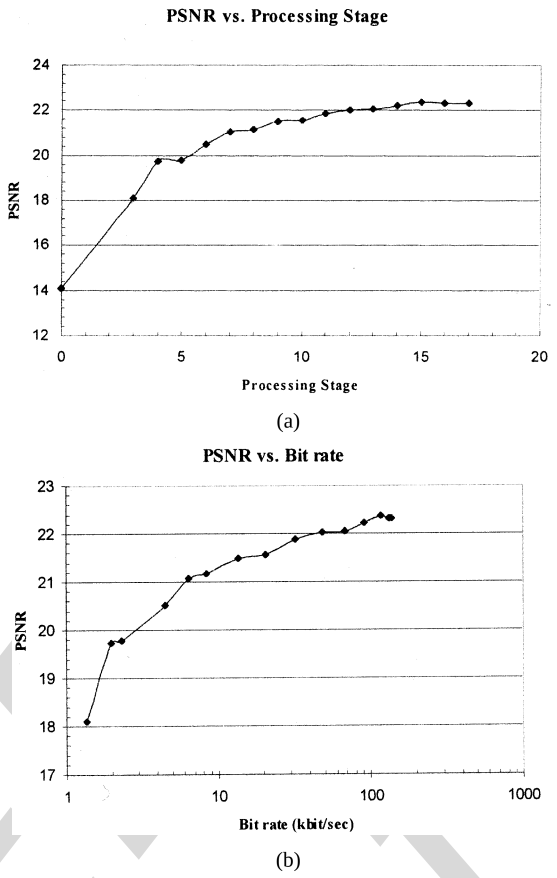


Fig. 10. (a) PSNR versus processing stage and (b) PSNR versus bit rate of the predicted image using Frame 93 and Frame 96 of the *mobile and calendar* sequence.

- Fig. 10(a) shows the convex-shaped PSNR versus processing stage plot. This suggests that the proposed technique codes the largest possible block of uniform motion at each stage. So the rate of improvement in PSNR decreases as the processing stage increases.

Similar analysis is shown in Fig. 11 for Frame 5 and Frame 6 of the *table tennis* sequence. It shows a better rate of PSNR improvement because the frames of the *table tennis* sequence used in the analysis have dominant translational moving objects where the proposed algorithm fits well.

Table III and Figs. 12 and 13 show the bit-rate performance of block-based motion estimation and the proposed technique with the *mobile and calendar* and *table tennis* sequences. The number of bits required for transmitting the motion vector information between two frames for the block-matching technique is calculated as follows.

- The actual motion vectors for each macro-block between two frames are found.
- Each motion vector x component is subtracted from the x component of the motion vector in the previous block (i.e., the one directly before it). Similarly, a differential code for the y component is found.
- The motion vector table for H.263 [12] is used to find the total number of bits required for differential codewords.

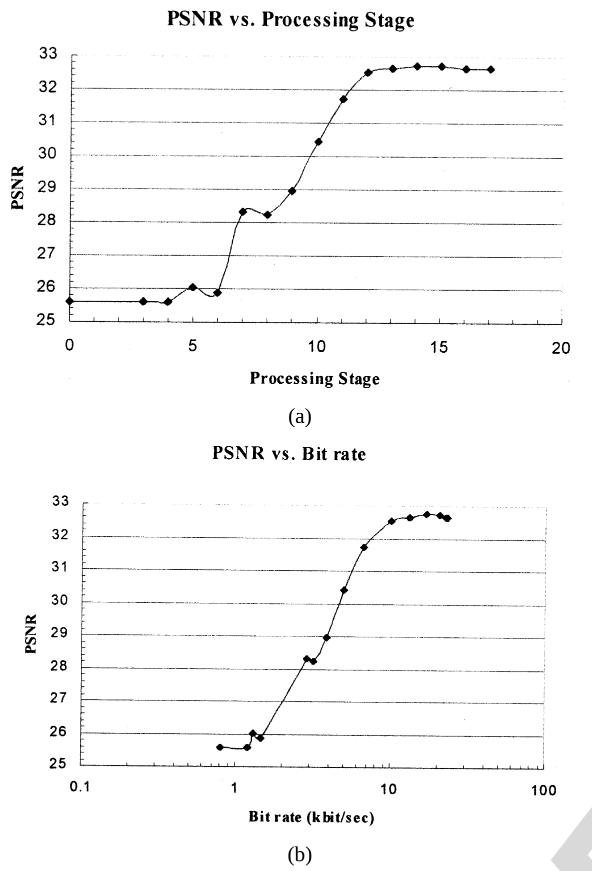


Fig. 11. (a) PSNR versus processing stage and (b) PSNR versus bit rate of the predicted image using Frame 5 and Frame 6 of the *table tennis* sequence.

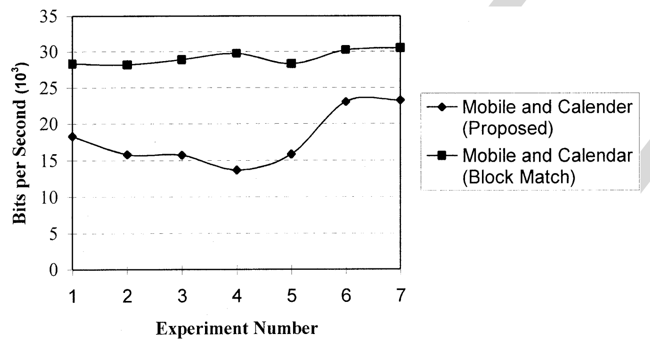


Fig. 12. Bit-rate comparison of block-matching technique and proposed technique for the *mobile and calendar* sequence.

Table III shows that the number of processing stages required in the proposed technique to get the predicted image of the same quality using the block-matching technique with block size (16×16) is nine or ten, which matches theoretically calculated values. Figs. 12, 13 show that the proposed technique requires significantly lower bit rates than block-based coding. Where other recent motion coding schemes have been compared on the same terms against block-based standards, they report improvement of prediction image PSNR of between 0.3 [13] and 0.6 dB [14]. At typical resolutions, this translates into a reduction in bit rate at constant prediction image PSNR of between 5% and 20%. The technique proposed here offers a reduction of about 30% in motion information bit rate.

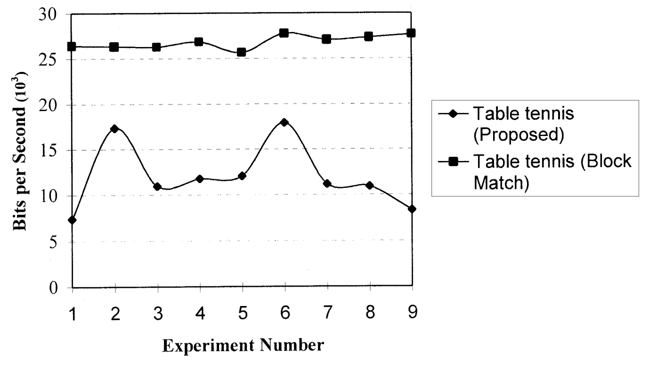


Fig. 13. Bit-rate comparison of block-matching technique and proposed technique for the *table tennis* sequence.

TABLE III
BIT-RATE PERFORMANCE WITH ABT TO ACHIEVE SAME-QUALITY PREDICTED IMAGE AS WITH THE BLOCK-MATCHING TECHNIQUE FOR THE *MOBILE AND CALENDAR* SEQUENCE

Exp. No.	Block Matching Technique (16 × 16 blocksize)		Proposed Technique				
	PSNR	Bit Rate (bps)	Processing Stage	PSNR	Number of transmitted regions (A)	Number of bits to transmit boundary info. (B)	Bit Rate (6+12×A+B) ×30 (bps)
1	22.14	28320	9	22.46	11	473	18330
2	22.19	28200	9	22.36	7	437	15810
3	22.14	28920	9	22.43	7	434	15720
4	21.44	29760	9	21.49	7	366	13680
5	21.54	28320	9	21.59	9	414	15840
6	21.14	30240	10	21.16	10	642	23040
7	20.83	30540	10	20.84	11	638	23280

V. CONCLUSION

We have proposed a global-to-local motion estimation algorithm, integrated into a video coder, that provides motion segmentation and efficient coding of moving object boundaries. The scheme produces qualitatively good segmentations of moving scenes and outperforms block-oriented standards for video compression. It provides fine-grain control of the tradeoff between motion information and residue information, so could be the basis for a rate-adaptive coder.

REFERENCES

[1] R. J. Clarke, *Digital Compression of Still Images and Video*. New York: Academic, 1996, pp. 256–259.
[2] S. C. Yoon, K. Ratakonda, and N. Ahuja, "Low bit-rate video coding with implicit multiscale segmentation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 9, pp. 1115–1124, Oct. 1999.
[3] A. Kaup, "Object-based texture coding of moving video in MPEG-4," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 9, pp. 5–15, Feb. 1999.
[4] J.-H. Moon, J.-H. Kweon, and H.-K. Kim, "Boundary block merging (BBM) technique for efficient texture coding of arbitrarily shaped object," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 9, pp. 35–43, Feb. 1999.
[5] S. H. Cho, R. C. Kim, S. S. Oh, and S. U. Lee, "A coding technique for the contours in smoothly perfect eight-connectivity based on two-stage motion compensation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 9, pp. 59–69, Feb. 1999.

[6] J. L. Baron, D. J. Fleet, and S. S. Beauhemin, "Performance of optical flow techniques," *Int. J. Comput. Vis.*, vol. 12, no. 1, pp. 43–77, 1994.

[7] H. T. Nguyen, M. Worring, and A. Dev, "Robust Motion-based Segmentation in Video Sequences," Univ. Amsterdam, Amsterdam, The Netherlands, Tech. Rep., 1999.

[8] J. Shi and J. Malik, "Motion segmentation and tracking using normalized cuts," presented at the Int. Conf. Computer Vision (ICCV), Bombay, India, Jan. 1998.

[9] M. A. Shamim and J. A. Robinson, "Modified binary tree for contour coding and its performance analysis," in *Proc. 3rd Int. Symp. Wireless Personal Multimedia Communications*, Bangkok, Thailand, Nov. 12–15, 2000, pp. 603–608.

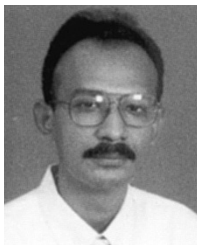
[10] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*. Reading, MA: Addison-Wesley, 1992.

[11] Y. Cohen, M. S. Landy, and M. Pavel, "Hierarchical coding of binary images," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-7, pp. 284–298, May 1985.

[12] Annex D of the ITU H.263+ Standard [Online]. Available: http://www.icsl.ucla.edu/~ipl/papers/AnnexD.pdf

[13] P. Hsu, K. J. R. Liu, and T. Chen, "A low bit-rate video codec based on two-dimensional mesh motion compensation with adaptive interpolation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 1, pp. 111–117, Jan. 2001.

[14] K.-W. Wong, K.-M. Lan, and V.-C. Siu, "An efficient low bit-rate video-coding algorithm focusing on moving regions," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, pp. 1128–1134, Oct. 2001.



Ahsan Shamim received the B.Sc. degree in electrical and electronics engineering from Bangladesh University of Engineering and Technology (BUET), Dhaka, in 1994, the M.Engg. degree in telecommunications engineering from Asian Institute of Technology (AIT), Bangkok, Thailand (under a FINNIDA scholarship) in 1996, and the M.Engg. degree in electrical engineering from Memorial University of Newfoundland, St. John's, NF, Canada, in 2000.

From 1997 to 1998, he was with the Telecommunications Program, AIT. Since then, he has been a Design Engineer working with DSP software for Nokia Mobile Phones (R&D), Salo, Finland. His research interests include digital signal processing, video coding, and image communications.

John A. Robinson (M'87) received postgraduate degrees in electronic engineering from The University of Essex, U.K., and in humanities from Memorial University of Newfoundland, St. John's, NF, Canada. **(AUTHOR: PLEASE SUPPLY DATES AND DEGREE TYPE—M.S./PH.D., ETC.)**

He is a Professor of Media Technology in the Department of Electronics, University of York, Heslington, York, U.K. , where he conducts research in image and video coding, wearable computers, and video-augmented environments. From 1996 to 2000, he held an Industrial Research Chair at Memorial University of Newfoundland. Previously, he was with Standard Telephones and Cables Ltd., Basildon, U.K., Bell-Northern Research Ltd., Verdun, QB, Canada, and The University of Waterloo, Waterloo, ON, Canada.

Dr. Robinson is a Fellow of the Institute of Electrical Engineers.