Variable size block matching motion compensation with applications to video coding

M.H. Chan, PhD
Y.B. Yu, MSc
A.G. Constantinides, PhD, CEng, FIEE

Indexing terms: Codes and decoding; Algorithms

Abstract: In block matching type motion compensation schemes, the image is divided into blocks of the same size. For each block a search is conducted in the previous frame to locate the best correspondence. For the scheme to succeed an implicit assumption has to be made that the motion within each block is uniform, an assumption which may not necessarily be correct, and as a result the quality of the prediction suffers. In the paper, a new motion compensation scheme based on block matching is presented, where the size for each block is variable. The proposed algorithm adaptively divides the image into blocks of variable size to meet the assumption on uniform motion for all blocks. The scheme has been successfully applied to simple interframe video coding. It is shown that the proposed algorithm can be extended to form the basis of a complete and efficient codec with low complexity. The possibility of the combination of the scheme with novel hybrid coding techniques to form sophisticated systems with low bit-rate performance, that compare favourably with other existing schemes, is also demonstrated.

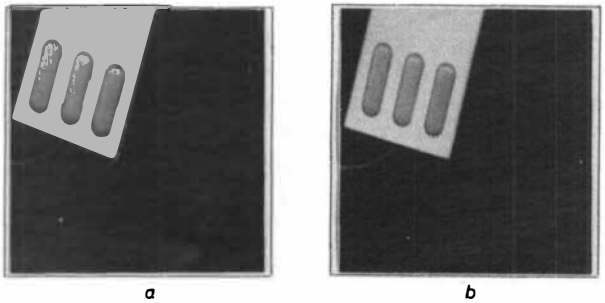
1 Introduction

The effectiveness of motion compensation in low bit-rate video coding is now widely recognised. In a typical interframe coding system, a prediction of the current frame is generated from the previous frame(s). The prediction error, which is much less correlated than the original picture, can then be coded at a much lower rate. At the receiving end the same prediction is generated from previously reconstructed frame(s), and is combined with the received prediction error to produce a reconstruction of the current frame. The key point to the success of such schemes is the ability to predict the current frame based on the previous ones. A good predictor produces a prediction which is very close to the true picture and, in the ideal case, results in an exact prediction which renders the transmission of error unnecessary. The simplest possible predictor is one that uses just the previous frame as the prediction. Such a predictor will only be ideal when the sequence of images is stationary. In many applications we are concerned with images of moving objects, and many of the changes that occur from frame to frame can be accounted for by motion. A much improved prediction can therefore be obtained if the motion vector for each pixel, or set of pixels, is estimated and the appropriate compensation made.

At present, block matching type algorithms are commanding more interest than other motion compensation techniques. Jain and Jain [1] originally proposed the technique which involves the division of the image into small fixed size blocks. For each block of pixels a search is conducted, within a confined window in the previous frame, to locate a best matching block. The relative displacement of the two blocks is then taken to be the motion, or displacement vector, and transmitted to the receiver as side information, which is then used to generate the same prediction at the receiving end. Because of its relative simplicity and effectiveness, the technique is adopted by the CCITT study Group XV in their various reference models for a video-conferencing codec which operates at $m \times 384$ Kbit/s^{-1} [2, 3], for which hardware has been built [4, 5].

Despite its increasing popularity, block matching motion compensation (BMMC) has its drawbacks. As the motion vectors are estimated on a block by block basis, there is an implicit assumption that the motion within each block is uniform. To maintain the validity of such an assumption, relatively small blocks (8×8 or 16×16) are used in practice. However, a decrease in the size of the blocks means that the number of blocks, and hence the number of motion vectors that have to be transmitted, increases, which results in a large overhead in side information.

To illustrate the situation, Fig. 1 shows pictures of a computer mouse and a coin. In the second picture, the



a *b*

Fig. 1 *Two consecutive frames with movement*
a mouse in rotational movement and
b coin in translational movement

Paper 7354I (E4, E14), first received 24th February 1989 and in revised form 15th March 1990

Dr. Chan is with Telecom Australia Research Laboratories, Customer Services and Systems Branch, 770 Blackburn Road, Clayton 3168, Victoria, Australia

The other co-authors are with the Signal Processing Section, Department of Electrical Engineering, Imperial College, London SW7 2BT, United Kingdom

mouse is slightly rotated in an anti-clockwise direction and the coin is moved to the right. Block matching motion compensation is used to predict the second picture from the first. Figs. 2*a* and 2*b* shows the partition of the image into fixed size blocks of 16 × 16 and 8 × 8 pixels respectively, and Figs. 2*c* and 2*d* show the corresponding results of prediction. The size of the search

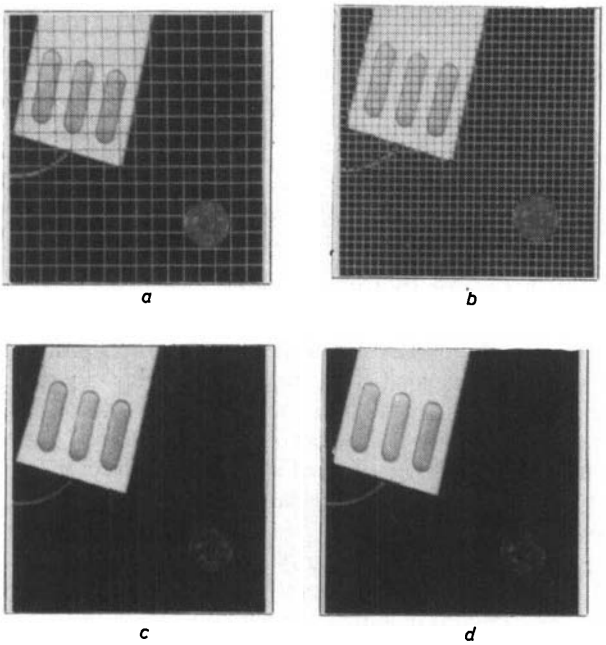


Fig. 2 *Block matching motion compensation: large blocks against small blocks*

a partition: 16 × 16
b partition: 8 × 8
c prediction: 16 × 16
d prediction: 8 × 8

window is ±7 pixels, and each motion vector can be coded with eight bits. For the case of 16 × 16 blocks, 256 motion vectors are coded, and 2048 bits are required. It can be seen that the result of prediction for the mouse is very poor, because the blocks are too big. The edges are jagged, and the lower left corner is not compensated because of the combined effect of large block size and limited search window. For the case of 8 × 8 blocks, the prediction is much more accurate, although the block size is still too big for the lower left corner of the mouse. There are now a total of 1024 blocks, which implies that 8192 bits are required to code the motion vectors, obviously too many for very low bit-rate video coding applications.

2 Variable size block matching motion compensation (VSBMMC)

The algorithm is essentially a split/merge segmentation scheme based on regular decomposition of an image into blocks of varying sizes, each of which has more or less uniform motion parameters. In an algorithmic sense this is similar to ordinary quad-tree segmentation of images [6], where the image is segmented into regions of uniform intensity. The decomposition can be binary (bin-tree decomposition) or quaternary (quad-tree decomposition). In the following discussion and in all subsequent simulations bin-tree decomposition is used, unless otherwise stated.

The current frame is initially split into a certain number of blocks by cutting the image alternately in the horizontal and vertical directions. For each block, an attempt is made to locate a block in the reconstructed previous frame which gives the best correspondence to the current block. The error of the approximation, given by the sum-square-error (SSE), is then used to indicate whether the block should be further split. The splitting goes on until the error of approximation for the block falls below a preset quality threshold, or a prescribed minimum block size has been reached. In this way a binary tree structure emerges, with leaf nodes which correspond to blocks of varying sizes, each of which can be described with a limited error by a block in the reconstructed previous frame. After that, neighbouring blocks under the same intermediate nodes are checked for possible merges, to establish whether the single merged block can be adequately approximated by a block in the reconstructed previous frame. Fig. 3 shows an example decomposition and its corresponding binary tree structure.

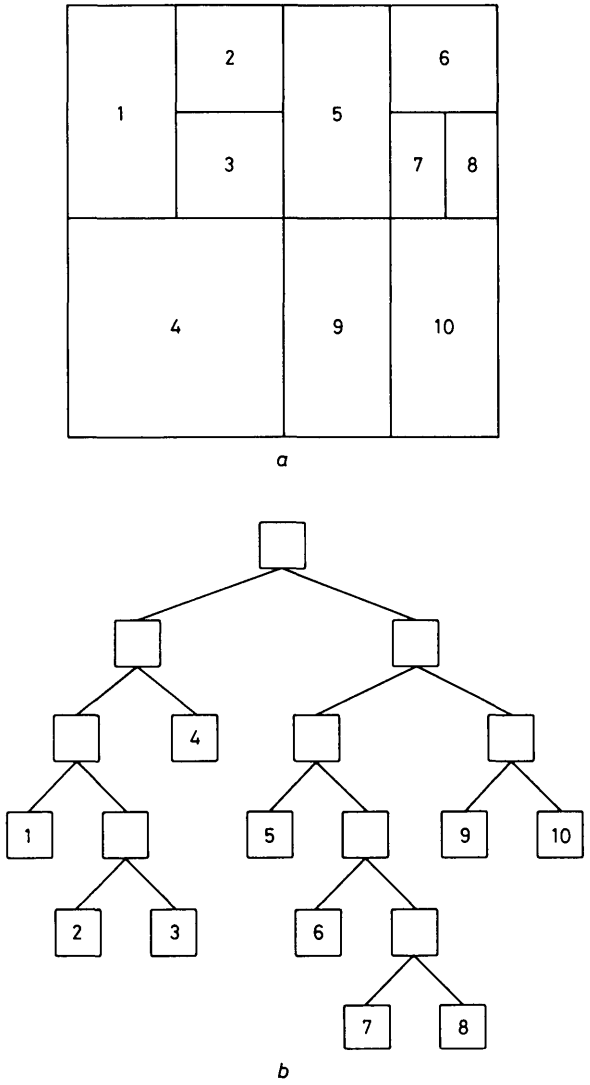


Fig. 3 *Variable size block matching motion compensation*

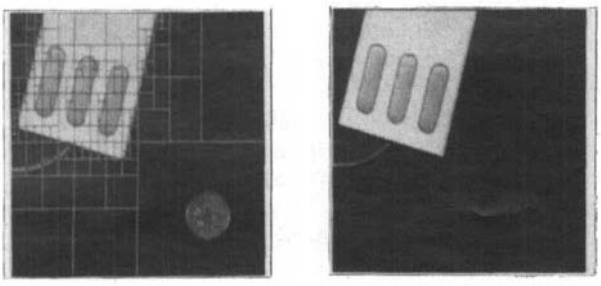
a an example of a decomposition
b resulting bin-tree

The coding requirement can be easily calculated from the number of blocks, or leaf nodes, in the final decomposition. For such a tree, with N leaf nodes, there must be $N-1$ intermediate nodes. For each leaf node one motion vector for the corresponding block is transmitted, which requires m bits. Note that m depends only on the size of the search window. In order to have the sizes and locations of the blocks available to the decoder, it is necessary to encode the binary tree as well. This requires one bit for each node, a total of $2N-1$. Therefore, the total coding requirement is $Nm + 2N - 1$ bits.

One obvious reason for the necessity of the merging operation is that the decomposition does not start at the top level. If this is so then at the beginning there will be a single block which corresponds to the entire image, and the search for a corresponding block in the reconstructed previous frame is certainly a waste of effort, unless the sequence is static at that particular instant. In most cases it would be desirable to start at an intermediate level, so that the initial decomposition is reasonably close to the final decomposition, and therefore reduces the amount of computation. The other reason for merging is that the division of a block (e.g. one which after all cannot be motion compensated) into smaller ones does not necessarily result in better approximation. Therefore, it is necessary to check for possible merges bottom-up to preserve the available bits for coding. For a thorough discussion see Reference 7.

The search for corresponding blocks is conducted in a logarithmic manner similar to that by Koga *et al.* [8], which is a modification of the original scheme proposed by Jain and Jain [1]. The method is called coarse-fine three steps search. As the name implies, the search is carried out in a coarse to fine manner within a search window of ± 7 pixels, and requires a total of 27 error calculations, as opposed to 225, if an exhaustive search is used. Computation is significantly reduced when the blocks are large. However, as the splitting proceeds, blocks become smaller, and the search is increasingly likely to be trapped in local optima. Therefore, in practice, a block size is specified below which an exhaustive search is made.

The effectiveness of variable size block matching is seen when the scheme is applied to the same pictures shown in Fig. 1. Fig. 4*a* shows the decomposition. It can be seen that while the coin, which is undergoing pure translation, and the most part of the background are described by large blocks, the mouse which is undergoing rotation is described by smaller blocks, especially near the edges. Fig. 4*b* shows the result of the prediction which



a *b*

Fig. 4 *Result of prediction which uses VSBMMC*
a partition
b prediction

can be directly compared with those in Fig. 2. While the quality of the prediction exceeds that shown in Fig. 2*d*, the coding requirement is far less, as shown in Table 1.

Table 1: Fixed against variable size block matching

	fixed 16 × 16	fixed 8 × 8	variable
SNR, dB	22.20	25.83	28.37
Coding requirement, bits	2048	8192	1449

Signal to noise ratio (SNR) is calculated as

$$10 \log \frac{\sigma_s^2}{\sigma_e^2} \quad (1)$$

where σ_s^2 is the variance of the original image and σ_e^2 is the variance of the error image.

As far as computational requirement is concerned, there is a penalty to be paid for the use of variable size block matching. For the example shown in Fig. 3, the computation required will roughly be twice as much as that for fixed size block matching, the exact figure depends on where the decompositoin process starts. The worst case is when we proceed top down through the tree, i.e. starting with the engire image as one single block, in which case the computation will be roughly four times as much. For a more detailed analysis see Reference 7.

From a large number of simulations on different sequences, it is estimated that on average the computation required is two to three times as much as compared with fixed size block matching.

In actual hardware implementation, it makes much sense to pose an upper, as well as a lower bound, on the size on the blocks for various reasons, and the amount of computation will be reduced, without sacrificing too much picture quality.

3 Application to simple inter-frame video codecs

The proposed scheme can be used to replace the more traditional motion compensation block in any interframe coder, for example the one proposed in Reference 1 where the prediction error is coded by the discrete cosine transform (DCT). A block diagram is shown in Fig. 5.

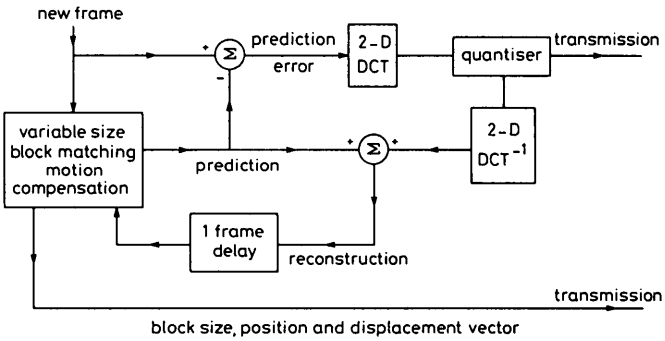


Fig. 5 *Simple interframe coder which incorporates VSBMMC*

The DCT used is an adaptive scheme, based on Chen and Smith [9], which classifies each block into one of four classes, each with a different bit allocation scheme.

Simulations are performed to compare the new scheme with the more traditional one which is based on fixed size block matching.

In this, and all subsequent simulations, three different video sequences, of a woman, a split screen and a man are used. They are all typical video-conferencing sequences, with different degrees of motion involved. The pictures are in quarter CSIF (common source input format) with a resolution of 144 × 176. Only the luminance component (8 bits per pixel) is coded. Fig. 6 shows for each sequence a typical frame in the relatively active region. The target frame rate is set at 15 frames/s^{-1}, and the coding rate is 64 Kbit/s^{-1}.

Fig. 7 shows the results in terms of SNR coding the three sequences using the scheme shown in Fig. 5 with both fixed and variable size block matching. The block size is set at 16 × 16 for the fixed size case. Fig. 8 shows for each sequence a typical reconstruction which corresponds to those shown in Fig. 6, for both fixed and variable size block matching.

Improvements are observed from all simulations with variable size block matching. However, even with VSBMMC, at 64 Kbit/s^{-1}, the quality of the reconstructions is still far from satisfactory, given the complexity. In the next section, the algorithm in a modified form is proposed as a complete coder, thereby the need for the DCT block is eliminated. The results show superior quality compared with the simple interframe DCT codec, while the complexity is reduced by the elimination of the error coding stage.

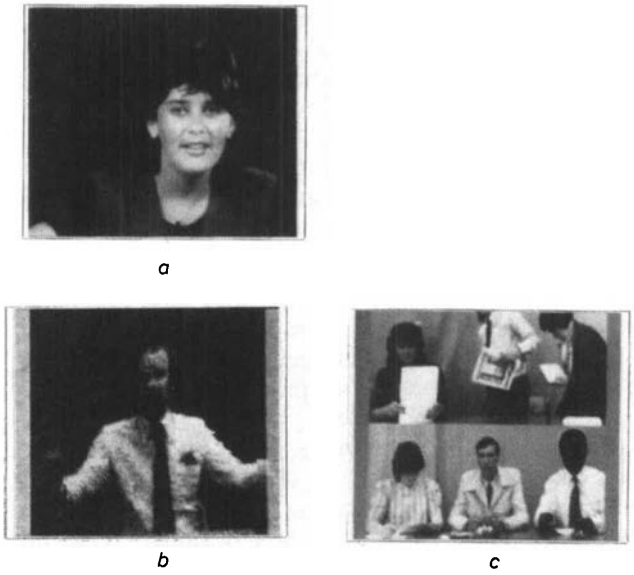


Fig. 6 *Typical original frames from the test sequences*
a woman
b man
c split screen

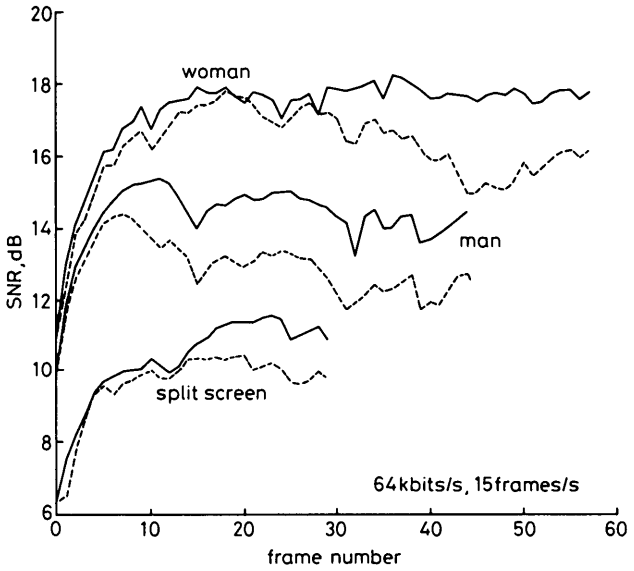


Fig. 7 *SNR comparison: BMMC against VSBMMC in interframe coding*

4 Intra/inter-frame block segmentation coding (IIBSC)

What we have proposed in Section 2 is, in essence, motion compensation by segmentation. Instead of having a fixed partition of the image and then try to find a motion vector for each region, we segment the image into regions of uniform motion. In the present case the segmentation is guided by regular decomposition. This is analogous to segmented image coding, where an image is partitioned into regions of uniform intensity. While the idea of motion compensation is to exploit temporal redundancy within a video sequence, intensity segmentation is for the exploitation of spatial redundancy within an image. These two ideas can be naturally combined into one, which forms the basis of a new methodology: video coding by hybrid intensity/motion segmentation. Each incoming frame is partitioned into regions of either uniform intensity or uniform motion parameters. In effect, regions in an image which can be satisfactorily approximated by similar regions in the previous frame are motion compensated (interframe coding), while others are intraframe coded by intensity segmentation.

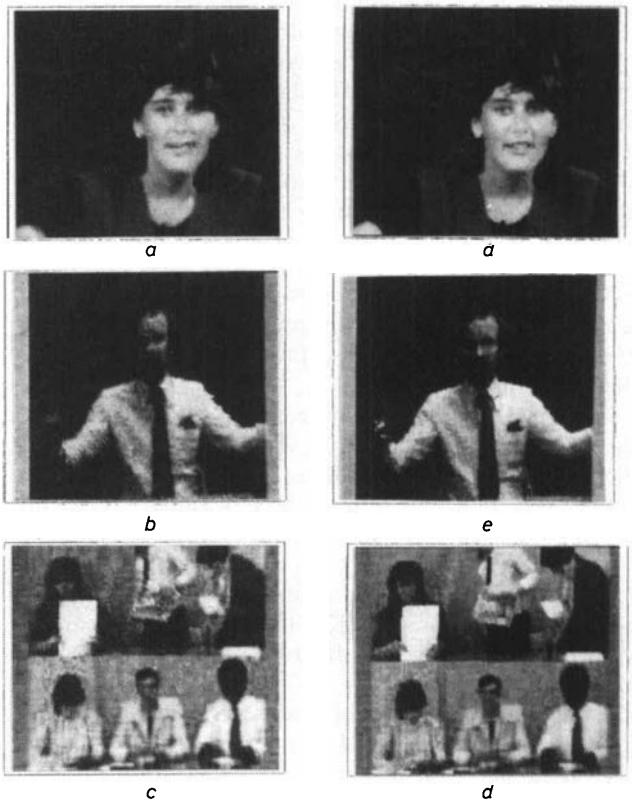


Fig. 8 *Typical reconstructions at 64 Kbit/s^{-1}: BMMC against VSBMMC in interframe coding*
a, b and *c*: BMMC + DCT *d, e* and *f*: VSBMMC + DCT

The current frame to be approximated, regarded as a two dimensional function $f(x, y)$, where $(x, y) \in I$, I being the picture plane, is initially partitioned into n rectangular blocks $S_1, S_2, \ldots, S_n$ by cutting the picture plane I alternately in the horizontal and vertical directions. Let $f_i(x, y)$ designate the pixel values of block S_i, $f_i(x, y) = f(x, y)$ for $(x, y) \in S_i$. Block S_i can be approximated by its mean intensity value $\bar{f}_i$,

$$f_i(x, y) \approx \bar{f}_i = \frac{1}{m_i} \sum_{(x, y) \in S_i} f(x, y) \tag{2}$$

where m_i is the size of the block S_i, with an error of approximation given by $SSE_{\text{intra}}(S_i)$,

$$\text{SSE}_{\text{intra}}(S_i) = \sum_{(x, y) \in S_i} (f(x, y) - \bar{f}_i)^2 \tag{3}$$

Alternatively the block S_i can be approximated by another block $r_i(x, y)$ extracted from the reconstruction of the previous frame $\hat{f}^0(x, y)$,

$$f_i(x, y) \approx r_i(x, y) = \hat{f}^0(x - \tilde{d}_x, y - \tilde{d}_y)$$
$$\text{for } (x, y) \in S_i \tag{4}$$

where $(\tilde{d}_x, \tilde{d}_y)$ is the motion vector which minimises the error of approximation,

$$\mathrm{SSE}_{\mathrm{inter}}(S_i) = \min_{(dx,\,dy)} \sum_{(x,\,y)\,\in\,S_i} (f(x, y) - \hat{f}^0(x - d_x, y - d_y))^2 \quad (5)$$

The overall SSE of the block, $\mathrm{SSE}(S_i)$, is then taken to be the minimum of the two sum-square-errors,

$$\mathrm{SSE}(S_i) \equiv \min(\mathrm{SSE}_{\mathrm{intra}}(S_i), \mathrm{SSE}_{\mathrm{inter}}(S_i)) \quad (6)$$

and is used as a measure which indicates whether the block S_i is adequately represented. If $\mathrm{SSE}(S_i)$ is above a prescribed quality threshold SSE_{th}, the block is further split into two. This process is repeated recursively for all blocks, until $\mathrm{SSE}(S_i) \leqslant \mathrm{SSE}_{th}$ for all i. In this way a binary tree structure emerges, with leaf nodes which correspond to blocks of varying sizes, each of which can be described with a limited error, either by a block in the previous reconstructed frame (interframe blocks), or by its local mean intensity value (intraframe blocks). Then neighbouring leaf nodes under the same intermediate nodes are checked for possible merges. If the merging operation results in no increase in SSE, or the SSE after the merge is below the threshold, the merge is effected.

The algorithm behaves like a bin-tree encoding scheme when the current frame cannot be approximated by the previous one, for example, when a blank previous frame is the starting point. On the other hand, when the current frame is a replica of the previous one, the entire image is represented by a single block with a motion vector of zero. If the current frame can be adequately described by the previous one, with different motion vectors in different parts of the image, a binary tree representation is generated to describe the various parts. If there exist parts of an image that cannot be described by motion, these are coded in an intraframe manner by bin-tree intensity segmentation.

While the use of SSE to control the split/merge operation seems natural, other possibilities have been investigated. In particular the use of maximum-absolute-error (MAE) is shown to give better results, and is thus adopted in the following simulations. For more details refer to References 7 and 10.

Fig. 9 shows the first four resulting frames coding the man image at 64 Kbit/s^{-1} using the scheme just presented. Fig. 10 shows again, for each sequence, the reconstruction which corresponds to those shown in Figs. 6 and 8. Fig. 11 plots the results in terms of SNR.

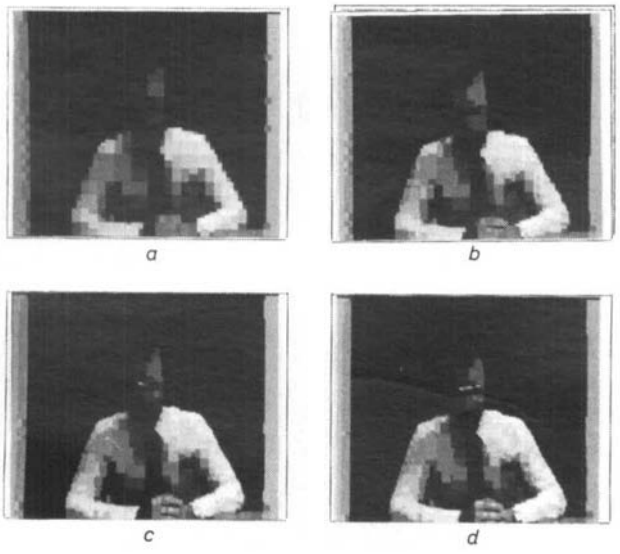


Fig. 9 *IIBSC: first four frames reconstructed at 64 Kbit/s^{-1}*

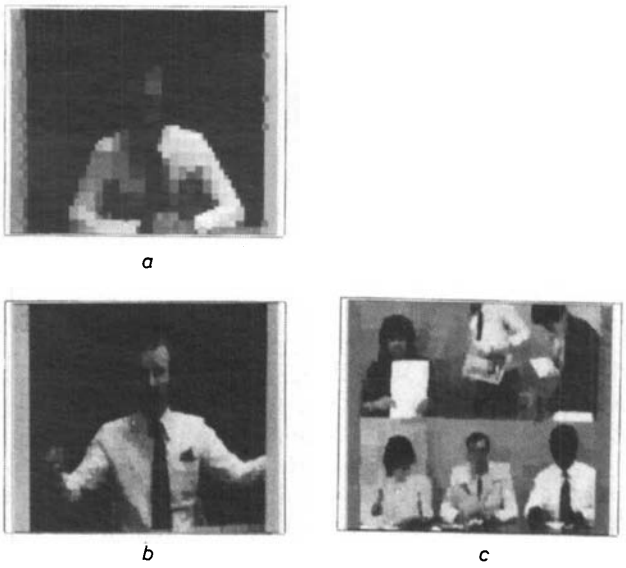


Fig. 10 *IIBSC: typical single frames reconstructed at 64 Kbit/s^{-1}*

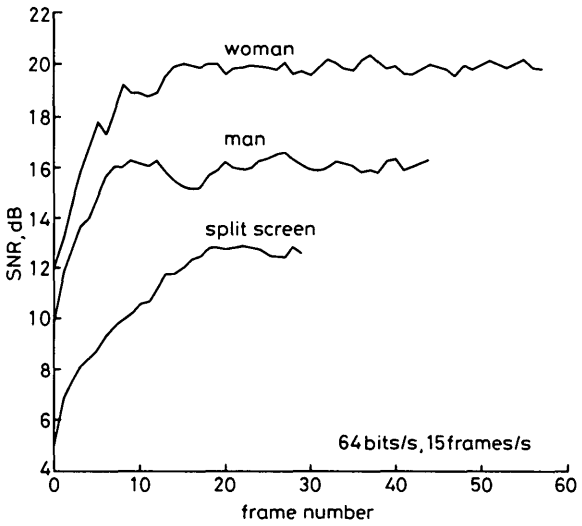


Fig. 11 *IIBSC: reconstructions at 64 Kbit/s^{-1}, SNR plot*

Note that the resulting pictures are much clearer in comparison with those in Section 3, and the edges are much sharper. For simple sequences like the woman image the results are quite acceptable, whereas those for the split screen image look artificial and are similar to 'paint-by-number' pictures.

Although the result obtainable at such a low rate may not be in a form suitable for immediate application, the relative simplicity of the scheme, especially at the decoding side, cannot be overlooked.

If higher quality is desired at the expense of complexity, the scheme can be further modified and combined with sophisticated techniques which results in a highly efficient codec, as reported in the following section.

5 Applications to more sophisticated systems: intra/inter-frame block segmentation transform coding (IIBSTC)

The scheme described in the previous section is characterised by the fact that the prediction error, as a result of the hybrid segmentation process, is completely ignored. Depending upon how efficient the prediction error can be dealt with, it might be advantageous to code and transmit it explicitly so as to retain some feeling of naturalness. The interframe DCT coder presented in Section 2 is

in a certain way such a scheme. Its inferior performance, as evident from the simulation results, is a result of the following facts.

(i) No intraframe coding is involved. An interframe signal in many cases is not necessarily easier to code.

(ii) The quantisation and bit allocation strategy, though optimal in the SSE sense, involves the transmission of a variance matrix for each frame. At low bit-rate this overhead becomes very significant, and better results can be obtained if the variances are modelled rather than transmitted.

The coder proposed in the recent CCITT standard H.261 for video-conferencing at $m \times 384$ Kbit/s^{-1} [11], and related reference models, encapsulates these two important points. In their scheme the motion compensation is performed by fixed size block matching. Transmitted with each block is an attribute which describes the type of the block, that is either motion-compensated (MC on) or non motion-compensated (MC off), inter or intraframe, and coded or non-coded. Note that 'MC off' is a special case of 'MC on', with a motion vector of (0, 0) which is not transmitted. The coded/non-coded switch is included so that blocks with very little AC energy are not coded. The DCT transformed coefficients, if the block is to be coded, are then quantised by a uniform quantiser followed by a variable length coding (VLC) seheme. Note that *a priori* knowledge about the distribution of the coefficients is used in the design of the VLC. Although not optimal, this eliminates the need to transmit the variance matrix, and reduces the complexity of hardware implementation. Rate control is accomplished by varying the step size of the uniform quantiser on a GOB (group of blocks, defined as two consecutive horizontal lines of blocks)-by-GOB basis. For more details see References 2 and 3.

Efficient though it is, at 64 Kbit/s^{-1}, it would be desirable to have better results, given the complexity. One way to improve matters is again to replace the motion compensation block by the more efficient approach based on variable size block matching. However, this creates impairments in the sense that the blocks with which motion vectors are associated (the result of a variable partition) do not coincide with the blocks on which the DCT is performed (the result of a fixed partition). This leads to a second thought on why a picture is partitioned, and in particular why into fixed size blocks, for transformation. The answer to the first part of the question is for adaptivity and ease of implementation. With the picture divided into smaller blocks, the signal contained within each block can be considered to be stationary, and well established statistical techniques can therefore be applied. Adaptivity is achieved with the allocation of a different number of bits to different blocks, dependent upon a measure of activity for the block. The choice of block size has long been shown to be a compromise between several factors. The use of smaller blocks results in higher adaptivity, but the correlation among blocks cannot be exploited, and therefore limits the compression ratio achieved. The use of larger blocks can better exploit the picture correlation as a whole, but the stationary assumption within each block may then be violated, and the quality as a result suffers. Therefore, we are faced once more with a situation analogous to what is encountered in block matching motion compensation, and the same strategy of the use of variable block size can be applied.

5.1 *Variable size transform (VST)*

A recent study [12] demonstrates the feasibility of an entirely new class of transform techniques, the variable block size transform (VBST), whereby the picture is partitioned into blocks of varying size. The criterion for the partition is that the signal contained in each resulting block is essentially stationary, and can be dealt with by more or less the same number of bits in achieving the same reconstruction quality. This results in a partition which in effect follows the edges of the picture, with a larger concentration of blocks in highly non-stationary regions. The same number of bits is then allocated to all blocks, which results in a variable-size/constant-rate adaptive transformation scheme, as opposed to the traditional fixed-size/variable-rate schemes.

5.2 *The combination of VBST with VSBMMC*

Note that the resulting hybrid prediction error of IIBSC, and its implicit partitioning is well suited to the application of VBST, as all the blocks are of essentially the same SSE. A block diagram of the proposed scheme which combines the two techniques is shown in Fig. 12. The

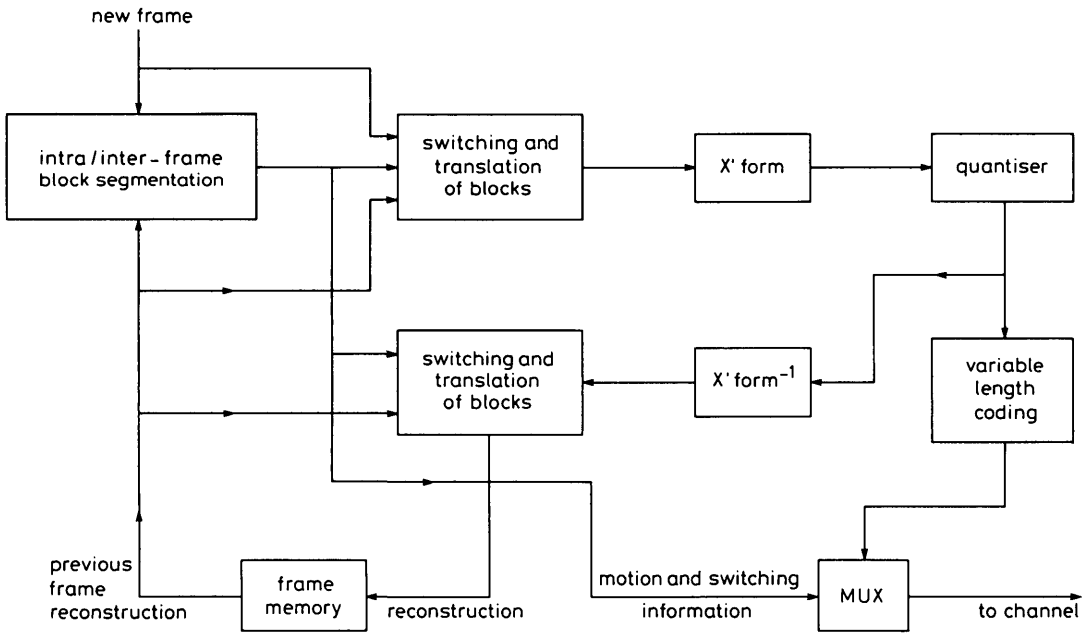


Fig. 12 *Inter/intra-frame block segmentation transform coding*

overall structure of the system is very similar to that proposed in Reference 11. The major difference is that the blocks can be of varying size.

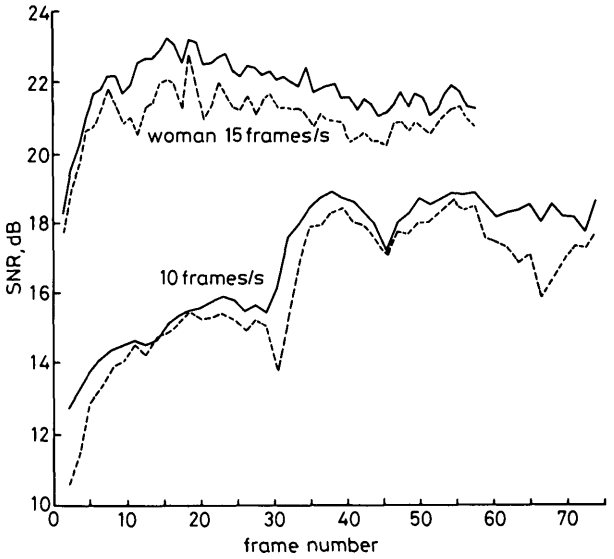


Fig. 13 *IIBSTC against CCITT RM6: reconstructions at 64 Kbit/s^{-1}, SNR plot*

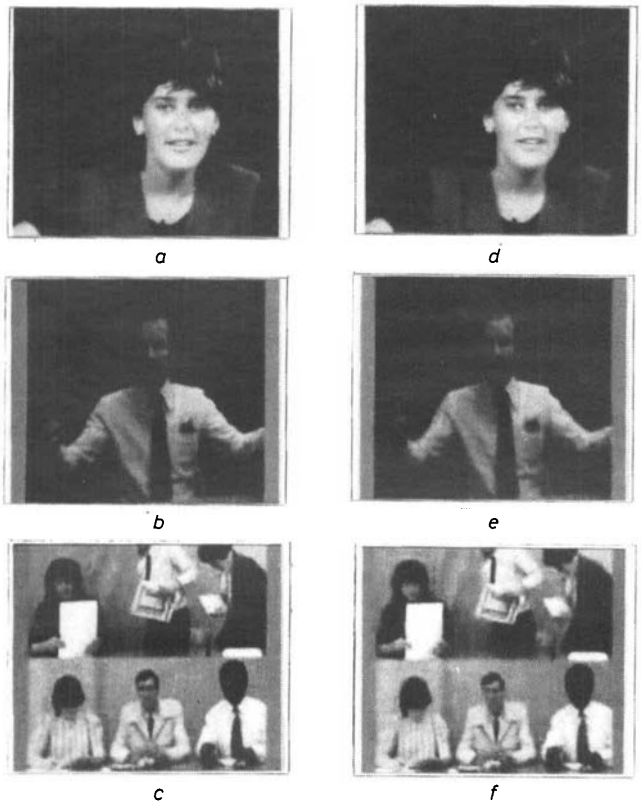


Fig. 14 *IIBSTC against CCITT RM6: typical reconstructions at 64 Kbit/s^{-1}*

a, *b* and *c*: IIBSTC
d, *e* and *f*: CCITT RM6

(i) Each incoming frame is first intra/interframe segmented as in IIBSC, with several modifications.

(*a*) The splitting of blocks is done in a quaternary manner rather than binary, since from our experience square blocks are more suited for transformation.

(*b*) No bottom-up merging is done. Recall that the reason for merging is to eliminate splits which result in no error reduction. In the current case the primary concern is a partition where each block is of comparable SSE.

(*c*) Instead of recursively splitting a block until all SSEs are below the prescribed threshold, the number of resulting blocks is explicitly controlled by always splitting the block with the largest SSE, until a prescribed number of blocks has been reached.

(ii) For each block the DCT is then performed, either on the original data, if it is an intraframe block, or on the motion compensated frame difference, if it is an interframe block. The resulting coefficients are quantised using a uniform quantiser defined by a step size g. The quantised coefficients are then scanned and transmitted sequentially in a zig-zag manner. Low frequency components are transmitted first, followed by increasing high frequency components up to the last non-zero quantised coefficient. The string of quantised coefficients are then coded with a two-dimensional variable length coding (2D VLC) scheme. This part of the coder is very similar to the one proposed in Reference 3.

(iii) Rate control is achieved by varying the quantiser step size on a block-by-block basis, rather than on a GOB-by-GOB basis, as in the CCITT proposal, as in our case the concept of GOB no longer applies.

For a more thorough description of the scheme, see Yu, Reference 13.

5.3 Simulation results

Simulations have been carried out to assess the effectiveness of the proposed codec. The target coding rate is 64 Kbit/s^{-1}. For the woman image the frame rate is again 15 frames/s^{-1}. The split screen image and the man image are combined into one single sequence, and the frame rate is set at 10 frames/s^{-1}. For comparison purposes the results of RM6 which operates at 64 Kbit/s^{-1} are plotted alongside those obtained by the proposed scheme in Fig. 13 for the two sequences. In all cases improvements can be observed using the proposed scheme. Fig. 14 shows some typical reconstructions with the use of both methods. The results obtained with the use of RM6 appear blurred because of the use of an interloop filter [3], without which the results are worse. For IIBSTC the interloop filter is not necessary.

6 Conclusions

A method of motion compensation, based on variable size block matching, has been described. In comparison with traditional fixed size block matching schemes, better results are obtained. Fewer overhead bits are required by the new scheme to produce comparable quality in prediction, and improved predictions are generated at comparable bit-rates. The concept of motion compensation by segmentation is introduced. While regular decomposition is one method for its implementation, other possibilities are currently being explored.

When applied to video coding, a new degree of freedom is created in the sense that the degree of motion compensation carried out, and thereby the proportion in bit-rate taken up by the motion compensation stage and subsequent error coding stages, can be sensibly controlled. In operation at very low rates, most schemes which directly act on the prediction error, if they are not sufficiently efficient, result in noisy reconstructions, as in the case with the coder described in Section 3. The coder presented in Section 4 introduces the concept of hybrid intensity/motion segmentation as a means towards video coding. This coder produces clean and pleasing pictures at very low rates. An additional desirable feature of the proposed scheme is that the complexity of the system is

relatively low. This is mainly a result of the removal of the error coding stage.

When the above scheme is combined with variable block size transform coding technique (which maintains its efficiency even at a very low rate), highly efficient video-conferencing codecs can be constructed.

The above techniques are useful for video coding, but other possible applications in motion segmentation and analysis are currently under study.

7 References

1 JAIN, J.R., and JAIN, A.K.: 'Displacement measurement and its application in interframe coding', *IEEE Trans.*, **COM-29,** (12), 1981, pp. 1799–1808

2 CCITT: 'Specifications for reference model version 2 (RM2)'. SG XV, Specialists Group on Coding for Visual Telephony, Doc. 141, 1986

3 CCITT: 'Description of reference model 6 (RM6)'. SG XV, Specialists Group on Coding for Visual Telephony, Doc. 396, 1988

4 PARKE, I., and MORRISON, D.G.: 'A hardware motion compensator for a videoconferencing codec'. Proc. IEE Colloquium on Motion Compensated Image Processing, 1987, pp. 1/1–1/5

5 DE VOS, L., STEGHERR, M., and NOLL, T.G.: 'VLSI Architectures for the full-search blockmatching algorithm'. Proc. ICASSP '89, **M5.22,** May 1989, pp. 1687–1690

6 HOROWITZ, S.L., and PAVLIDIS, T.: 'Picture segmentation by a tree traversal algorithm', *J. Assoc. Comput. Mach.*, **23,** 1976, pp. 368–388

7 CHAN, M.H.: 'Image coding algorithms for video-conferencing applications'. PhD Thesis, 1989, Imperial College, University of London

8 KOGA, T., *et al.*: 'Motion compensated interframe coding for video conferencing'. Proc. Nat. Telecommun. Conf., 1981, New Orleans, LA, pp. G5.3.1–G5.3.5

9 CHEN, W.H., and SMITH, H.: 'Adaptive coding of monochrome and color images', *IEE Trans.*, **COM-25,** (11), 1977, pp. 1285–1292

10 CHAN, M.H., and CONSTANTINIDES, A.G.: 'Inter/intra-frame block segmentation coding for video signals'. 1989 Internat. Symposium on Computer Architecture and Digital Signal Processing, CA-DSP '89, **1,** 1989, Hong Kong, pp. 563–567

11 CCITT: 'Codec for audiovisual services at $n \times 384$ Kbits/s'. Fascicle III.5, Rec. H.261, 1988, Melbourne

12 YU, Y.B., and CONSTANTINIDES, A.G.: 'Variable block size and position transform coding'. Fourth European Signal Processing Conference, **2,** Grenoble, France, 1988, pp. 1011–1013

13 YU, Y.B.: 'Transform coding of images based on shape adaptive models'. PhD Thesis, 1990, Imperial College, University of London