New Fast Algorithms for the Estimation of Block Motion Vectors

Bede Liu, *Fellow, IEEE*, and André Zaccarin, *Member, IEEE*

***Abstract*—Block motion estimation using exhaustive search is computationally intensive. Previously proposed fast algorithms reduce the computation by limiting the number of locations searched. This is sometimes done at the expense of the accuracy of the estimates and can result in motion-compensated prediction images with an appreciably higher mean squared error (MSE) than that obtained by exhaustive search. We present two algorithms for block motion estimation that produce similar performance to that of exhaustive search but with computation reduced by a factor of 8 or 16. The algorithms are based on motion-field and pixel subsampling. We first determine a subsampled motion field by estimating the motion vectors for a fraction of the blocks. The motion vectors for these blocks are determined by using only a fraction of the pixels at any searched location and by alternating the pixel subsampling patterns with the searched locations. We then interpolate the subsampled motion field so that a motion vector is determined for each block of pixels. The new algorithms are more robust than previously proposed fast algorithms. Both algorithms can easily be combined with a hierarchical search strategy. One of our algorithms is fully compatible with the Motion Picture Expert Group Standard (MPEG-I).**

I. Introduction

THE coding of video sequences has been the focus of a great deal of research in recent years. High-definition television (HDTV), videoconferencing and CD-ROM archiving are some of the better known applications. The high redundancy that exists between the successive frames of a video sequence makes it possible to achieve high compression ratios. A natural way to exploit the redundancy between consecutive frames consists of using frame $n-1$ to predict the next frame of the sequence, i.e., frame n. Frames n and $n-1$ are often referred to as the present and previous frames. The technique of block motion estimation is currently favored by many researchers in the field. In this technique, the present frame of the sequence is divided into rectangular or square blocks of pixels. For each block in the current frame, we look for the block of pixels in the previous frame that is the closest to it, according to a predetermined criterion. This closest block is then used as a predictor for the present block. The relative positions of the two blocks define a motion vector associated with the present block. The collection of all motion vectors defines a motion field and is sent to the receiver.

Among the various criteria that can be used as a measure of the match between the two blocks [1], the mean absolute difference (MAD) is favored because it requires no multiplication and gives similar performance as the mean squared error (MSE). For the present frame n, we denote the intensity of the pixel with coordinates (i,j) by $F_n(i,j)$. We refer to a block of $M \times N$ pixels by the coordinate (k,l) of its upper left corner. The MAD between the block (k,l) of the present frame and the block $(k+x, l+y)$ of the previous frame can then be written as

$$MAD_{(k,l)}(x,y) = \frac{1}{MN} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} |F_n(k+i, l+j) - F_{n-1}(k+x+i, l+y+j)|. \quad (1)$$

The motion vector $v(k,l)$ of the block (k,l) is given by

$$v(k,l) = \arg\min_{(x,y)} \mathrm{MAD}_{(k,l)}(x,y) \quad (2)$$

where the vectors $(k+x, l+y)$ are valid block coordinates.

For each location in the previous frame to be tested, the calculation of the MAD requires $2MN$ additions. In most cases, it is possible to bound the maximum displacement that an object can move between two frames, thus limiting the search to a reduced area. If a maximum displacement of w pixels/frame is allowed for both the x and y directions, there are $(2w+1)^2$ locations to search for the best match to the present block.

The technique known as exhaustive search computes the MAD at the $(2w+1)^2$ locations of the search area to find the optimal motion vector. To perform motion estimation in real time, the number of operations required by the exhaustive search is often too high. To reduce the computational complexity, a number of fast search algorithms have been proposed [2]–[7]. These algorithms reduce the number of computations by limiting the number of locations searched to find the motion vector. These techniques rely on a monotonically increasing MAD around the location of the optimal vector to iteratively determine that location. The MAD surface, however, of-

Manuscript received February 25, 1992; revised August 5, 1992. This work was supported by a Sony graduate fellowship and by the Office of Naval Research and the Naval Research Laboratory under Grant N00014-89-J-1327. This paper was recommended by Associate Editor Yrjö Neuvo.

B. Liu is with the Department of Electrical Engineering, Princeton University, Princeton, NJ 08544.

A. Zaccarin is with the Department of Electrical Engineering, Université Laval, Québec, Canada G1K-7P4.

IEEE Log Number 9207595.

1051-8215/93$03.00

ten has several minima in which these algorithms can be trapped. This will lead the algorithm to a final motion-vector estimate with a higher MAD than that of the optimal vector. In this paper, we present alternatives to reduce the computational complexity of block-based motion estimation. Instead of limiting the number of locations searched, we either reduce the number of allowed motion vectors or the number of pixels used to estimate the motion vectors. Two fast algorithms are obtained by combining these two techniques. Our algorithms perform nearly as well as the exhaustive-search technique, yet they have a computational complexity comparable to those of the fast-search algorithms previously reported. Our algorithms can be easily combined with a hierarchical-search strategy. One of the fast algorithms is also fully compatible with the Motion Picture Expert Group Standard (MPEG-I).

Section II presents a technique to estimate a motion vector for each block of pixels by using only a fraction of the pixels in the block. Section III describes a technique to estimate the motion vectors that reduces the computation by 50%. This technique first determines the motion vector of half of the blocks by any method and then estimates the motion vectors of the other half of the blocks by a limited search. In Section IV, we present a technique to achieve sub-block accuracy by first dividing each block into four sub-blocks, then finding a motion vector for one of the sub-blocks, and finally interpolating the motion field to get an estimate of the motion vectors for all sub-blocks. In Section V, we combine these techniques to obtain two algorithms that reduce the computation to 1/8 or 1/16 of that using full search, with a performance close to that of the full search.

II. Pixel Decimation

When matching a block from the present frame to a block from the previous frame, the matching criterion is usually evaluated using every pixel of the block. Since block matching is based on the assumption that all pixels in a block move by the same amount, a good estimate of the motion could be obtained, in principle, by using only a fraction of the pixels in a block. If too few pixels are used, however, there will eventually be a reduction in the accuracy of the motion estimates. In this section we present a pixel-decimation technique using pixel subsampling by a factor of 4 that preserves the motion-estimation accuracy.

A. Description of the Technique

Subsampling of pixel blocks as a means of reducing computational complexity of the block-matching algorithms has been previously reported in the literature. Bierling [8], [9] introduced a hierarchical motion-estimation technique in which a first approximation of the motion field is obtained from low-pass-filtered and subsampled images. Previously, Koga *et al.* [2] proposed subsampling of unfiltered images by a factor of 2 and noted an increase in the prediction-error entropy by 0.04 bit/pixel. A 4 to 1 subsampling ratio increased the error entropy by 0.1 bit/pixel and was considered excessive. As in [2], the pixel-decimation technique we present in this section uses unfiltered images.

Fig. 1 shows a block of 8×8 pixels with each pixel labeled **a**, **b**, **c**, or **d** in a regular pattern. We call pattern **A** the subsampling pattern that consists of all the **a** pixels. Similarly, patterns **B**, **C**, and **D** are the subsampling patterns that consist of all the **b**, **c**, and **d** pixels, respectively.

If only the pixels of pattern **A** are used for block matching, then the computation is reduced by a factor of 4. However, since 3/4 of the pixels do not enter into the matching computation, the use of this subsampling pattern alone can seriously affect the accuracy of the motion vectors. To reduce this drawback, we propose using all four subsampling patterns, but only one at each location of the search area and in a specific alternating manner.

Fig. 2 shows some pixels forming part of the search region in the previous frame. The pixels are labeled **1**, **2**, **3**, and **4** in a regular pattern. The labeling of the pixels refers to which of the four subsampling patterns of Fig. 1 is to be used for computing the matching at that location. That is, when computing the match at locations labeled **1** (i.e., when the upper-left pixel of the block to match falls on those locations), pattern **A** is used. Similarly, pattern **B**, **C**, or **D** is used when computing the match at locations labeled **2**, **3**, or **4**.

In other words, if pattern **A** is used at the location (x, y), then it is also used at locations $(x + 2i, y + 2j)$ for i, j integers within the search area; pattern **D** is used at location $(x + 1 + 2i, y + 2j)$, pattern **B** at $(x + 2i, y + 1 + 2j)$, and pattern **C** at $(x + 1 + 2i, y + 1 + 2j)$. For each of the subsampling patterns, we obtain a motion vector that minimizes the MAD over the locations where that pattern is used. We then compute the match for each of the four vectors, but using all the pixels. The one that has minimum MAD among the four is selected as the motion vector for the block.

Alternating between the four subsampling patterns allows us to use all the pixels of the block within four different search locations. Furthermore, this specific way of alternating the subsampling patterns will use all pixels of the previous frame within the search area. These two facts reduce the possibility of overlooking one-pixel-wide horizontal, vertical, or diagonal lines. It should be noted that of the six possible alternating patterns, only two have the properties described above. One is shown in Fig. 2. In the other, **3** is to the right of **1**, and **4** is below **1**.

Each of the four patterns in the procedure just described has a subsampling factor of 4 to 1. It is possible to devise a similar procedure, but with a subsampling factor of 2 to 1. In such a procedure, one has only to deal with two different subsampling patterns.

B. Computational Complexity

Let L be the number of locations searched by a block-matching algorithm, and let K be the number of operations per pixel needed to compute a given error criterion. Thus, a total of LKN^2 operations is needed to estimate the motion vector of a block of $N \times N$ pixels. Computa-

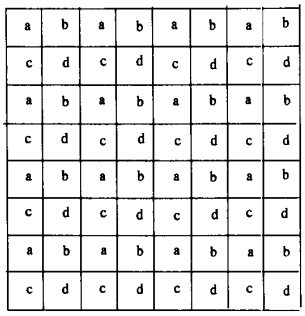


Fig. 1. Patterns of pixels used for computing the matching criterion with a 4 to 1 subsampling ratio.

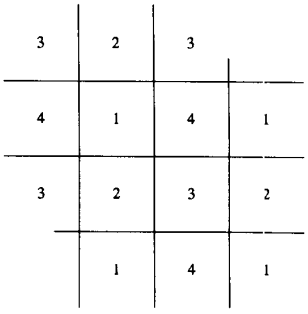


Fig. 2. Alternating schedule of the four pixel subsampling patterns over the search area.

tionally, the technique using a subsampling factor of 4 requires $LK(N^2/4)$ operations per block to determine the four motion vectors corresponding to the four subsampling patterns. An extra $3KN^2$ operations are needed to complete the computation of the matching criterion over all the pixels of the block for these four motion vectors. Since $3KN^2 < LK(N^2/4)$, this overhead is negligible. Overall then, the scheme described in Section II-A reduces the computational complexity of full block exhaustive search by a factor of 4. Similarly, the technique of using a subsampling factor of 2 requires 50% of the operations needed by an exhaustive full search.

C. Results

To test the performance of this approach, simulations were run on two sequences. The first is a 10-s segment of the Tennis sequence, and the second is a 7-s segment of the Football sequence. Both sequences are in the CIF format (30 frames/s with 352 × 240 pixel/frame). These sequences offer an interesting combination of still, slow- and fast-moving objects, camera zoom and panning, and objects with relatively different sizes. Although the simulations were run on all the frames of both sequences, we only present the results obtained with the first 90 frames of the Tennis sequence since they are representative of the results obtained throughout the two sequences.

We used a maximum allowed displacement of seven pixels in the x and y directions ($w = 7$) with a block size of 8 × 8 pixels. The error criterion is the MAD, and we compare the algorithms using the MSE per pixel of the motion-compensated frames.

Fig. 3 shows the performance of a standard 4 to 1 subsampling and that of the proposed approach. The performance of the exhaustive-search algorithm without any subsampling is also shown. It is seen that the proposed approach has an MSE very close to the exhaustive search using no subsampling. It is significantly better than the standard 4 to 1 subsampling (without alternating) that can have an MSE per pixel as much as 50% higher than that of the full exhaustive search.

Simulations were also run for a 2 to 1 subsampling ratio. The performance improvement obtained by alternating two subsampling patterns was less significant. This is especially true for large block sizes, such as 16 × 16.

III. Subsampled Motion-Field Estimation

Motion fields of image sequences are usually smooth and slowly varying, with discontinuities limited to the boundary of objects moving in different directions or with different velocities. Even when motion vectors are assigned to a block of pixels, these characteristics of the motion field can be observed. As a result, it is not uncommon to find neighboring blocks with identical or nearly identical motion vectors. This characteristic of the motion field is exploited in many schemes used to code the motion vectors [10], [11].

The technique proposed in this section also exploits this characteristic, but to reduce the computational complexity of block-based motion-estimation algorithms. We assume that the motion vector of a block of pixels is very likely to be similar to the motion vector of one of its neighboring blocks. We first obtain a subsampled motion field by estimating the motion vectors for a fraction of the blocks using any block-matching algorithm. That motion field is then appropriately interpolated so that a motion vector is determined for each block of pixels. The interpolation requires only a small number of operations, so a subsampling of the motion field by a factor of 2 reduces the computational complexity by the same factor. The cost of this reduction in computation is a reduction in the prediction quality of the present frame. The most serious degradation occurs when an object with the same dimension as the block size moves with a velocity completely different from those of its neighboring blocks. Our experience has shown that subsampling of the motion field by a factor of 2 causes only minimal increase in prediction error.

A. Description of the Technique

We first estimate the motion vectors for half of the blocks in a frame, as indicated by the darkened blocks of Fig. 4, by using any block-matching technique including full search or pixel decimation as described in Section II. The motion vectors of the white blocks are obtained from the vectors of the neighboring blocks as follows. Suppose block (a, b) is one of those white blocks. We simply test the four motion vectors of the adjacent black blocks

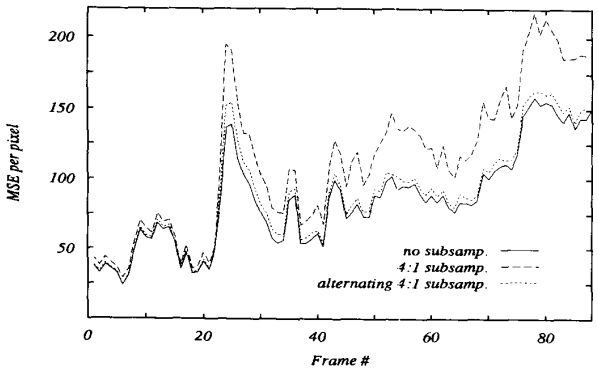


Fig. 3. MSE per pixel of the motion-compensated frames produced by exhaustive search with no subsampling, standard 4 to 1 subsampling, and alternating 4 to 1 subsampling patterns.

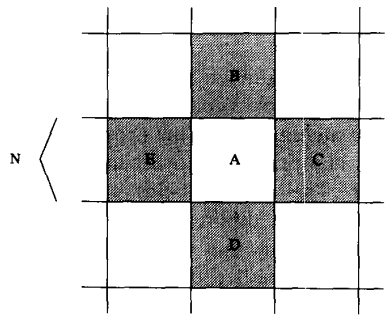


Fig. 4. Subsampling pattern used for the motion field.

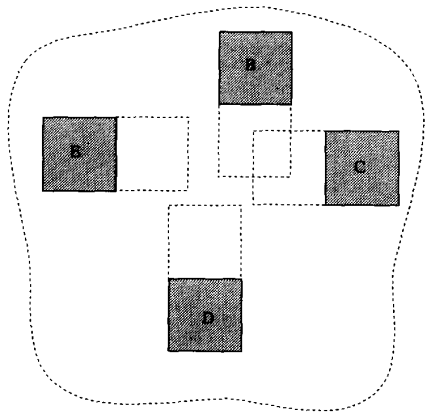


Fig. 5. Locations tested for best matching block of block **A**.

$(a \pm N, b \pm N)$ and choose the vector for which the MAD is the smallest (Fig. 5). For example, the motion vector assigned to block **A** is the motion vector of either block **B**, **C**, **D**, or **E**, whichever gives the smallest MAD.

By looking at Fig. 4, one can see why the proposed technique should perform nearly as well as a block-matching algorithms separately estimating a vector for each block. The motion of block **A** is very likely to be close to the motion of one of the surrounding blocks. If block **A** contains an object also contained in any of the other blocks, the motion estimate for block **A** will be accurate. If block **A** overlaps the edges of two objects moving in different directions, its motion estimate will be the motion of one of the two objects, as estimated by one of the neighboring blocks. This estimate will be no more inaccurate than an estimate of the motion directly using that block. The proposed technique does not perform well only when block **A** contains small objects whose dimensions are equal to or less than the block size and which are moving in different directions than the neighboring objects.

B. Computational Complexity

To quantify the computational complexity of this technique we assume, as in Section II-B, that the computation of the error criterion used by the search algorithm requires KN^2 operations for a block of $N \times N$ pixels for each of the L locations searched. Thus, a total of LKN^2 operations is needed to estimate the motion vector of a darkened block. Since a vector is calculated for every other block, an average of $(L/2)KN^2$ operations per block is needed. To this we add the number of operations needed to assign the motion vectors to the other half of the blocks for which four locations are tested. This step adds an extra $4KN^2$ operations for every other block. Overall, the proposed technique needs $(L/2 + 2)KN^2$ operations per block. In comparison, a full exhaustive-search algorithm, *i.e.*, an exhaustive search applied to every block, requires LKN^2 operations per block, so the proposed scheme requires roughly only 50% of the operations.

The number of bits required to code the motion field is also reduced roughly by 2, as only the motion vector of half of the blocks needs to be sent. In addition, every other block is assigned the motion vector of one of its four neighboring blocks. This information requires 2 bits for every such block or an overhead of only 1 bit/block. For a block size of 8×8 pixels, this is only $1/64$ bit/pixel. Therefore, besides reducing the computational complexity by half, the proposed technique also reduces the bit rate necessary to code the motion field.

C. Results

The simulation results we present in this section were also obtained with the Tennis and Football sequences, using the parameters and performance criterion described previously in Section II-C. We compare the performance of the new algorithm to that of the exhaustive-search block matching applied to all blocks (full exhaustive search). We also use an exhaustive search as the block-matching technique of the proposed algorithm. As before, the block size is 8×8 pixels.

Fig. 6 shows the results of the simulation for the first 90 frames of the Tennis sequence. For 88% of the frames, the increase in MSE relative to the full exhaustive search is less than 20%. Note that the increase in MSE is more significant after the fortieth frame of that sequence seg-

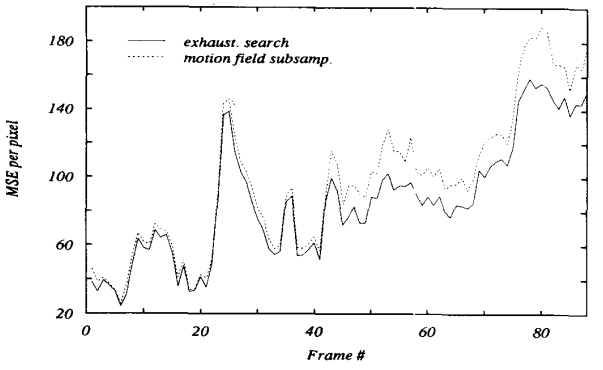


Fig. 6. MSE per pixel of the motion-compensated frames produced by the sub-sampled motion-field estimation and exhaustive-search techniques for the Tennis sequence.

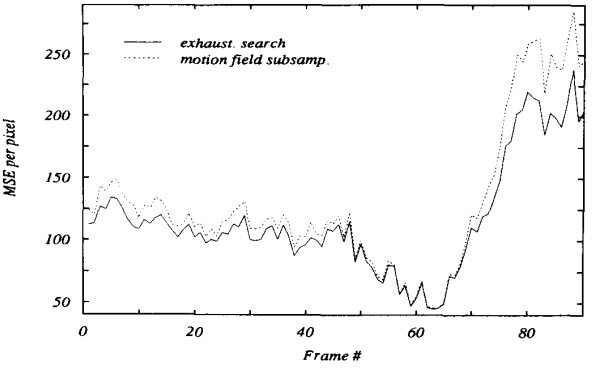


Fig. 7. MSE per pixel of the motion-compensated frames produced by the sub-sampled motion-field estimation and exhaustive-search techniques for the Football sequence.

ment. During those frames, the camera is zooming out so that the blocks at the border of the frames are equivalent to uncovered background blocks. For those blocks, the motion vectors found by using exhaustive search are usually uncorrelated with the true motion field. They also tend to point in different directions from one block to the next. In this case, the motion-field subsampling does not perform as well. However, it is interesting to note that the increase in MSE can be localized to a small number of blocks. For example, at the fifty-third frame, the MSE of the proposed approach is 25% higher than the MSE of the exhaustive search, but 86% of that MSE increase can be attributed to only 19 of the 1320 blocks. Furthermore, 95% of the blocks in this frame have an rms error per block of less than 5. Similar numbers are obtained for the other frames of that segment. These observations give strong support to the proposed approach as a viable alternative to exhaustive search for all blocks despite the local increase in MSE.

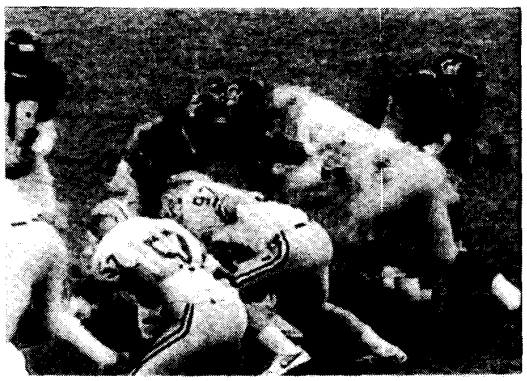
Fig. 7 shows the results of the simulation for the first 90 frames of the Football sequence. For this sequence, the increase in MSE relative to the full exhaustive search is less than 20% for 72% of the frames. As in the Tennis sequence, the increase is largely due to a small fraction of the blocks. For example, at frame 82, where the increase in MSE is 23%, 70% of the increase can be attributed to 58 blocks. Compared to the Tennis sequence, the number of blocks with larger MSE is higher. This is because there is a larger number of uncovered blocks due to the speed at which the action takes place and due to the number of objects moving in different directions. The blocks used by the full exhaustive search to predict these uncovered background blocks are the closest, in the MAD sense, to the original blocks. In most cases, however, they are not necessarily visually closer to the original blocks than the blocks used by the proposed approach. Fig. 8 shows a typical frame of the Football sequence (frame 82) and the predicted frames using full exhaustive search and the proposed technique. It is seen that, although the frame produced by the exhaustive search has a lower MSE, it is not visually better than the one produced by the proposed technique.

Fig. 9 shows the number of bits per frame needed to code the motion vectors produced by the full exhaustive-search block matching and by the proposed algorithm. For the proposed algorithm, the overhead of 1 bit/block is also included. The motion fields were coded using a Huffman code on the difference of adjacent motion vectors. These results show that the proposed technique does use, in general, a smaller number of bits to code the motion field. It is interesting to note that the segment, where the exhaustive search needs a larger number of bits to code the motion field, corresponds to the segment where the proposed technique has a higher MSE than the exhaustive search (see Fig. 7). The higher number of bits needed to code the motion field produced by the exhaustive search can be explained by the fact that neighboring motion vectors are not as correlated as they are elsewhere in the sequence. As we mentioned earlier, this lack of correlation causes a reduction in the performance of the proposed algorithm. The savings in the number of bits from the coding of the motion field by the proposed algorithm can be used to code the prediction error with a greater precision to compensate for the higher MSE. Similar observations were noted with the segments of the Tennis sequence.

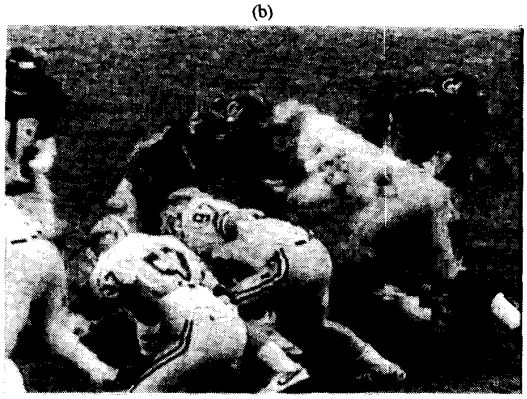
IV. Sub-block Motion-Field Estimation

Block-based motion estimation inherently assumes rigid body and translational motion of the entire block. Consequently, it performs poorly when two or more objects within the same block are moving in different directions or with different velocities. In these situations, a block is compensated with the motion vector of only one of the objects or with a vector not related to the motion of any of the objects. If smaller blocks are used, it is less likely that objects moving with different directions or with different velocities will be found within the same block. Furthermore, the rigid-body and translational-motion assumptions are more likely to be valid for smaller blocks.

(a)



(b)



(c)

Fig. 8. (a) Frame 82 of the Football sequence. (b) Predicted frame using exhaustive search. (c) Predicted frame using motion-field subsampling.

As a result, the quality of the motion-compensated prediction improves. The use of smaller block sizes, however, increases the bit rate because more motion vectors have to be sent to the receiver. There is an inherent tradeoff between block size and bit rate. In this section, we propose an approach for block-motion estimation that uses estimates made from smaller blocks, yet requires no increase in the number of motion vectors sent to the receiver. An overhead of at most 4 bit/block however, is needed. For comparable prediction quality, this approach reduces the computational complexity by a factor of 4.

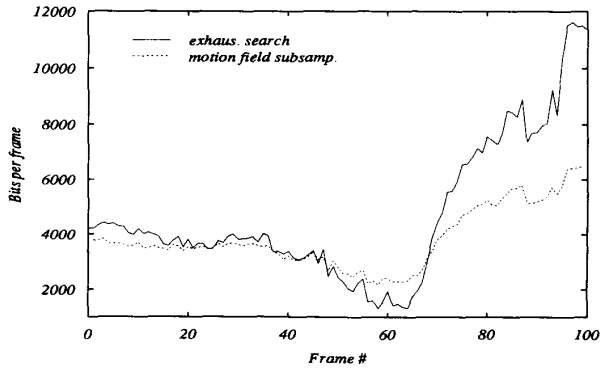


Fig. 9. Number of bits needed to code the motion vectors produced by the exhaustive-search block matching and the subsampled motion-field estimation (including overhead) for the Football sequence.

A. Description of the Technique

We assume as before that square blocks of $N \times N$ pixels are used by a full block-matching coding technique. Each block is divided into four sub-blocks of $N/2 \times N/2$ pixels, as depicted in Fig. 10. **A**, **a**, **b**, **c** are sub-blocks of a block, and **B**, **C**, and **D** are sub-blocks of neighboring blocks. We first determine the motion vector of one sub-block in each block, say the upper-left sub-blocks (**A**, **B**, **C**, **D**, ...). We then assign to the remaining sub-blocks the motion vector of one of the neighboring dark sub-blocks. Thus, sub-block **a** is assigned the motion vector of either sub-block **A** or **B** depending on whether the motion vector of **A** or **B** gives a better match; sub-block **b** is assigned the motion vector of either **A** and **C**; and sub-block **c** is assigned the motion vector of either **A**, **B**, **C**, or **D**.

Using this technique, the motion estimates for sub-blocks **A**, **B**, **C**, and **D** will obviously be more accurate than the ones obtained using the full block of pixels. Motion estimates for blocks **a**, **b**, and **c** will also be more accurate if those blocks are part of an object also covered by one of the blocks **A**, **B**, **C**, or **D**. This technique will not produce a good motion vector for objects confined to a small "white" sub-block for which a motion vector is not estimated in the first place. But the effectiveness of full search using the entire $N \times N$ block is also not clear in these cases. Our simulation results show that the proposed technique performs at least as well as a full block-matching algorithm.

B. Computational Complexity

To quantify the computational complexity of this technique, we assume, as before, that the error criterion used by the search algorithm requires KN^2 operations for a block of $N \times N$ pixels for each of the L locations being searched. To find a motion vector for a sub-block of $N/2 \times N/2$ pixels in each block, the number of required operations per block is $LKN^2/4$. To this, we add the number of operations needed to assign a motion vector to the other three sub-blocks of each block. There are, at

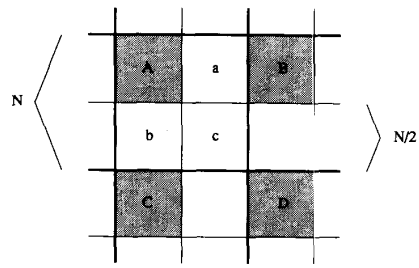


Fig. 10. Sub-block system for the new algorithm.

most, two locations for each of the sub-blocks **a** and **b** and four locations for sub-block **c**, for a total of eight locations. This represents an extra $8KN^2/4$ or $2KN^2$ operations. Overall, the number of operations per block is equal to $(L/4 + 2)KN^2$. This is approximately $1/4$ of the number of operations needed by a full block-matching algorithm.

Since this technique estimates only one new motion vector for each block of pixels, the total number of bits required to code the motion vectors is similar to the number of bits required by a full block-matching algorithm. However, we have to add the information needed to indicate to the receiver which of the motion vectors to assign to the three remaining sub-blocks of each block. The receiver must be informed how to choose between two vectors for two of the sub-blocks and among the four vectors for the other sub-block. This represents an overhead of 4 bit/block.

C. Results

We again used the Tennis and Football sequences with the parameters and performance criteria described previously to compare the proposed approach against full exhaustive-search block matching. The exhaustive search was also used as the block-matching algorithm for the proposed approach. Simulations were run for blocks of 16×16 pixels. This implies that sub-blocks of 8×8 pixels were used with the proposed algorithm.

Figs. 11 and 12 show the simulation results for the first 90 frames of the Tennis and Football sequences. As these figures show, the proposed approach performs better than a full exhaustive search, reducing the MSE per pixel by more than 10% for 42% of the frames of the Tennis sequence and for 24% of the frames of the Football sequence. As mentioned earlier, this sub-block approach requires only 25% of the computation of a full block-matching algorithm. The price paid is an overhead of 4 bit/block or, equivalently, only 0.016 bit/pixel for blocks of 16×16 pixels. Simulations using blocks of 8×8 pixels produced similar results, but the overhead is 0.0625 bit/pixel.

V. Fast Algorithms

So far in this paper, we have introduced three techniques that reduce the computational complexity of block

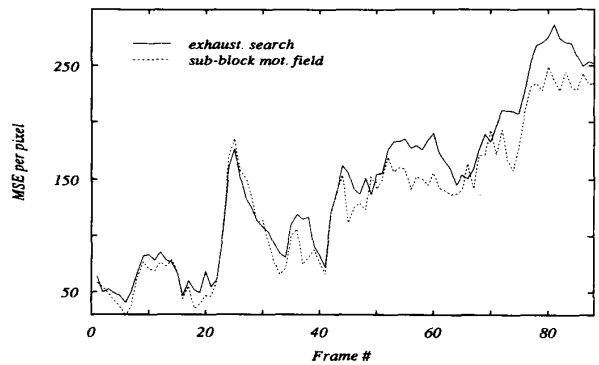


Fig. 11. MSE per pixel of the motion-compensated frames produced by the sub-block motion-field estimation and the exhaustive-search techniques for the Tennis sequence.

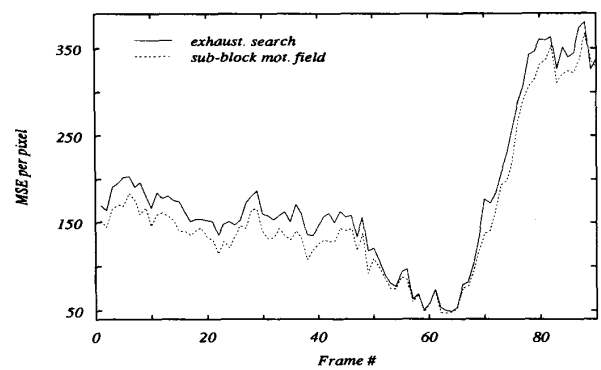


Fig. 12. MSE per pixel of the motion-compensated frames produced by the sub-block motion-field estimation and the exhaustive-search techniques for the Football sequence.

motion estimation by factors of 2 to 4. In this section, we combine the pixel-decimation technique using alternating patterns, presented in Section II, with the subsampled and the sub-block motion field-estimation techniques of Sections III and IV. By doing so, we obtain algorithms with complexity reduced by a factor of 8 to 16 from that of the full block exhaustive search. We will compare the performance of these fast algorithms to the performance of the three-step search (TSS) [2], a fast-search algorithm that limits the number of locations to be searched. Although the TSS does not visit the smallest number of locations to find the best matching block of pixels, it is probably one of the more robust algorithms since it visits all the locations considered by the other fast algorithms [3]–[7].

For each block of pixels, this algorithm visits 25 locations to estimate its motion vector. It allows for a maximum displacement of 6 pixels in the x and y directions. If KN^2 represents the number of operations needed to evaluate the matching criterion at one location for a block of N^2 pixels, the TSS requires only $25KN^2$ operations per block of pixels. The TSS algorithm can also be combined with a 2 to 1 subsampling of the block pixels. In that case, the algorithm requires only the equivalent of $12.5KN^2$

operations per block and, for blocks of 16×16 pixels or larger, the loss in performance is negligible.

A. Sub-block Motion-Field Estimation with Alternating Pixel-Decimation Patterns

Our first fast algorithm combines the sub-block motion-field estimation and the decimation technique using alternating pixel patterns. The first step of the algorithm consists of estimating, for each block of $N \times N$ pixels, the motion vector of a sub-block of $N/2 \times N/2$ pixels. Instead of using all the pixels of a sub-block, as in Section IV, those motion vectors are now estimated using the alternating subsampling patterns described in Section II. For the remaining sub-blocks, we assign them the motion vector of the neighboring sub-blocks that produces the minimum matching error, using all the pixels in those sub-blocks, as presented in Section IV.

The first step of the sub-block approach requires $LKN^2/4$ operations. With a pixel-decimation factor of 2, the number of operations is equal to $LKN^2/8$ plus $KN^2/4$. For a pixel-decimation factor of 4, the operation count is equal to $LKN^2/16$ plus $3KN^2/4$. The number of operations needed to assign a motion vector to the other sub-blocks is left unchanged and is equal to $2KN^2$ for both decimation factors. The total number of operations needed by this algorithm is then equal to $(L/8 + 2.25)KN^2$ for a decimation factor of 2 and equal to $(L/16 + 2.75)KN^2$ for a decimation factor of 4.

The performance of the proposed algorithm using alternating 4 to 1 subsampling patterns was compared to the performance of the TSS combined with a standard 2 to 1 subsampling. Simulations were run using blocks of 16×16 pixels and, for the proposed algorithm, a square searched area of 169 locations. This search area corresponds to maximum displacements of 6 pixels ($w = 6$) in either the x or y direction, which is the same maximum displacement that the TSS can estimate. Fig. 13 shows the results obtained using the first 90 frames of the Tennis and Football sequences. The performance of the full-block exhaustive search is also shown as a reference.

These simulation results show that our algorithm compares favorably to the TSS for similar computational complexity. The TSS algorithm, with 2 to 1 subsampling, requires $12.5KN^2$ operations while the new algorithm, for a pixel-decimation factor of 4 and $L = 169$, requires $13.3KN^2$ operations. Although our algorithm requires an overhead of 0.016 bit/pixel, it has the advantage of searching every location that an exhaustive search visits. Fast-search algorithms, like the TSS, make consecutive decisions to find the best motion vector. In some cases, it will happen that the location of the minimizing vector at one of the first steps does not reflect the actual location of the true minimizing vector. This will then lead the algorithm to a final motion-vector estimate with a higher MAD than that of the optimal vector. This phenomenon was observed in the last 100 frames of the Tennis sequence where the TSS MSE was twice as large as our algorithm's.

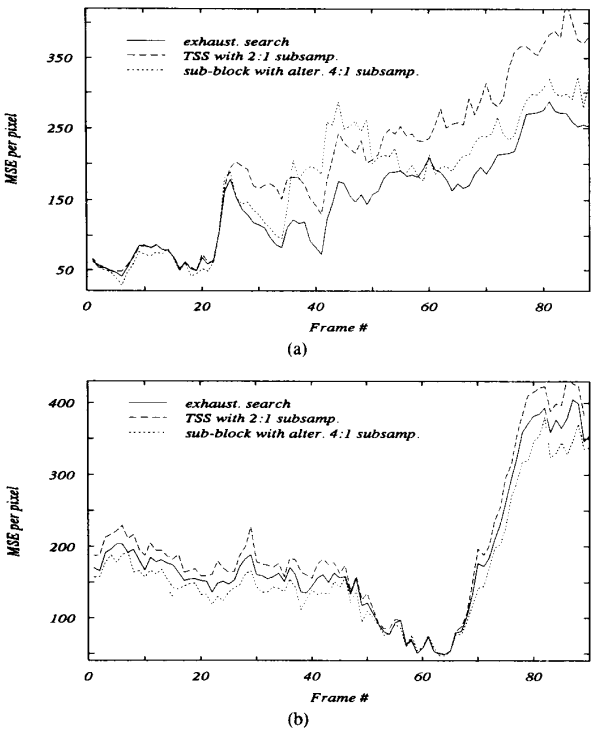


Fig. 13. MSE per pixel of the motion-compensated frames obtained using sub-block motion-field estimation combined with alternating 4 to 1 sub-sampling patterns, TSS combined with a standard 2 to 1 subsampling and full block exhaustive search (a) for the Tennis sequence and (b) for the Football sequence.

B. Subsampled Motion-Field Estimation with Alternating Pixel-Decimation Patterns

The algorithm presented in the previous section requires an overhead of 0.0625 bit/pixel when used with a block size of 8×8 pixels. In this section, we combine the subsampled motion-field estimation (Section III) with the decimation technique using alternating pixel patterns (Section II) to create a fast algorithm that can handle blocks of 8×8 pixels without requiring any overhead in the bit rate. The combination of the two algorithms is straightforward. The first step of the subsampled motion-field estimation, as described in Section III, consists of estimating a motion vector for every other block of pixels using any block-matching algorithm. For the fast version of the algorithm, those motion vectors are estimated using alternating 4 to 1 subsampling patterns. The second step of the subsampled motion-field estimation approach is left unchanged. The motion vectors of the remaining blocks are estimated using the neighboring block vectors without any pixel decimation. The complexity of the new algorithm is easily determined. The first step of the algorithm requires $(L/2)K(N^2/4)$ operations and the second step $2KN^2$ operations as before. The total number of operations is then equal to $(L/8 + 2)KN^2$.

As was done previously, we ran simulations to compare the performance of the proposed algorithm with the per-

formance of the TSS using blocks of 8×8 pixels. A square search area of 169 locations ($w = 6$) was used with the proposed algorithm. Fig. 14 shows the results of simulations. The performance of the full block exhaustive search is also shown as a reference. Once again, the performance of our fast algorithm compares favorably with the TSS for similar computational complexity. For $L = 169$, our algorithm requires $23.1KN^2$ operations compared with the $25KN^2$ operations needed by the TSS algorithm. The TSS could also be combined with a 2 to 1 subsampling of the pixels to further lower its complexity. When it is used with blocks of 8×8 pixels or smaller, however, the subsampling increases the sensitivity of the TSS to local minima of the MAD surface. This then leads the TSS to a higher number of erroneous decisions and higher MSE. These facts were also verified by simulations.

VI. Conclusion

In this paper, we have presented two fast algorithms that reduce the computational complexity of a full-exhaustive-search block matching by factors of 8 and 16. These algorithms are based on three techniques.

The first technique consists of alternating pixel-decimation patterns, thereby reducing the computational complexity of a full exhaustive search by a factor of 4 without significantly increasing the MSE of the motion-compensated frames. The subsampling patterns are alternated over the locations searched so that all pixels of a block contribute to the computation of the motion vector.

The second technique takes advantage of the smoothness of an image-sequence motion field. It starts with a subsampled motion-field estimation and then properly interpolates it so that a motion vector is determined for each block of pixels. Since the interpolation requires a small number of operations, a subsampling of the motion field by a factor of 2 reduces the computational complexity by the same factor.

The third technique, the sub-block motion-field estimation, reduces the computational complexity of standard block-motion-estimation schemes by a factor of 4. It first estimates a motion vector for each block of pixels using only one of its four sub-blocks.

Again, taking advantage of its smoothness, the motion field is interpolated so that a motion vector is assigned to each sub-block of pixels. The higher subsampling rate of the motion field is compensated by the increased accuracy of the motion-vector estimates that a smaller block size produces. This is achieved with a transmission overhead of 4 bit/block, which is negligible for blocks of 16×16 pixels.

Of the two fast algorithms, the more efficient one combines the sub-block motion-field estimation with decimation using alternating pixel patterns. Because of its required transmission overhead of 4 bit/block, it is better suited for blocks of 16×16 pixels or larger. The second fast algorithm, which requires no overhead, was obtained by combining the subsampled motion-field estimation with the same pixel-decimation technique. This algorithm is fully compatible with MPEG-I. Both algorithms can be easily combined with a hierarchical-search strategy.

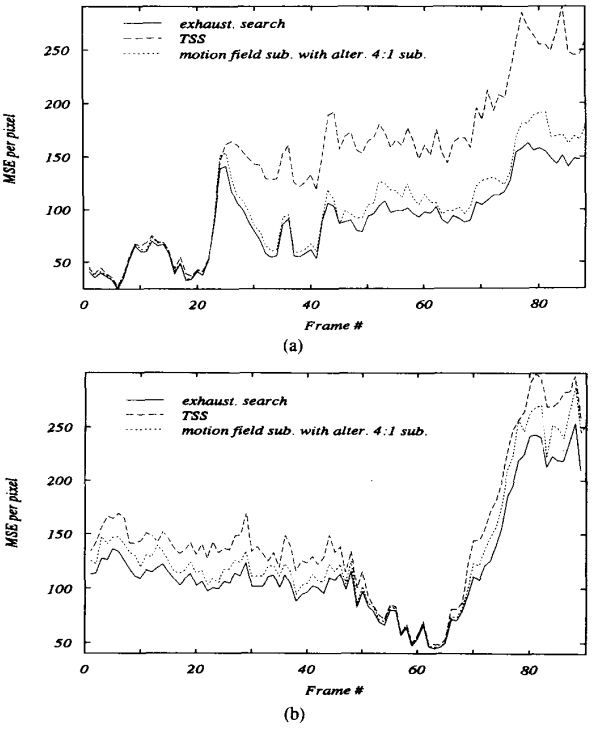


Fig. 14. MSE per pixel of the motion-compensated frames obtained using motion-field subsampling combined with alternating 4 to 1 subsampling patterns, TSS and full block exhaustive search (a) for the Tennis sequence and (b) for the Football sequence.

A number of fast algorithms for block motion estimation have been previously proposed. These algorithms reduce the number of computations by limiting the number of locations searched. They rely on a monotonically increasing MSE error around the location of the optimal motion vector to iteratively determine that location. In a number of cases, though, the MSE surface has several local minima in which these algorithms can be trapped. The algorithms we have introduced are more robust than these fast algorithms since every location of the searched area is considered in the estimation of the motion vectors used for the subsequent interpolation. Furthermore, by alternating the pixel-subsampling patterns with the searched locations, we minimize the effects of aliasing introduced by subsampling the pixels without prior low-pass filtering. Simulations showed that the performance of our algorithms compared favorably with that of the TSS for comparable computational complexity. In fact, our algorithms performed nearly as well as the exhaustive-search technique with only 1/8 or 1/16 of the computation.

References

[1] H. G. Musmann, P. Pirsch, and H. J. Gralleer, "Advances in picture coding," *Proc. IEEE*, vol. 73, no. 4, pp. 523–548, Apr. 1985.
[2] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, and T. Ishiguro, "Motion compensated interframe coding for video conferencing," in *Proc.*

Nat. Telecommun. Conf. (New Orleans, LA), Nov. 29–Dec. 3, 1981, pp. G5.3.1–5.3.5.

[3] J. R. Jain and A. K. Jain, "Displacement measurement and its application in interframe image coding," *IEEE Trans. Commun.*, vol. COM-29, no. 12, pp. 1799–1808, Dec. 1981.

[4] R. Srinivasan and K. R. Rao, "Predictive coding based on efficient motion estimation," *IEEE Trans. Commun.*, vol. COM-33, no. 8, pp. 888–896, Aug. 1985.

[5] S. Kappagantula and K. R. Rao, "Motion predictive interframe coding," *IEEE Trans. Commun.*, vol. COM-33, no. 9, pp. 1011–1015, Sept. 1985.

[6] A. Puri, H.-M. Hang, and D. L. Schilling, "An efficient block-matching algorithm for motion-compensated coding," in *Proc. Int. Conf. Acoust., Speech, Signal Processing*, 1987, pp. 25.4.1–25.4.4.

[7] M. Ghanbari, "The cross-search algorithm for motion estimation," *IEEE Trans. Commun.*, vol. 38, no. 7, pp. 950–953, July 1990.

[8] M. Bierling and R. Thoma, "Motion compensating field interpolation using a hierarchically structured displacement estimator," *Signal Processing*, vol. 11, no. 4, pp. 387–404, Dec. 1986.

[9] M. Bierling, "Displacement estimation by hierarchical block matching," in *SPIE, Visual Commun. Image Processing '88*, vol. 1001, pp. 942–951, 1988.

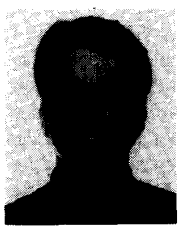
[10] M. T. Orchard, "A comparison of techniques for estimating block motion in image sequence coding," in *SPIE, Visual Commun. Image Processing IV*, vol. 1199, pp. 248–258, 1989.

[11] A. Wong, C.-T. Chen *et al.*, "MPIC: A video coding algorithm for transmission and storage applications," *IEEE Commun. Mag.*, vol. 28, no. 11, pp. 24–32, Nov. 1990.

Bede Liu (S'55–M'62–F'72) was born in Shanghai, China. He received the B.S.E.E. degree from National Taiwan University, Taipei, and the M.E.E. and D.E.E. degrees from the Polytechnic Institute of Brooklyn, Brooklyn, NY.

Prior to joining the Princeton University, Princeton, NJ, faculty in 1962, he had been with Bell Laboratories, the Allen B. DuMont Laboratory, and Western Electric Company. He has also been a Visiting Faculty Member at National Taiwan University, the University of California, Berkeley, Imperial College, London, and Jiao Tong University, Shanghai, China. He is Professor of Electrical Engineering at Princeton University. He edited the research volume *Digital Filters and the Fast Fourier Transform* (DHR, 1975) and coauthored the senior-level textbook *Digital Signal Processing: Theory, Design, and Implementation* (New York: Wiley, 1976). His current research interests lie mostly in image and signal processing.

Dr. Liu was an ADCOM member of the Acoustics, Speech, and Signal Processing Society, an ADCOM member of the Circuits and Systems Society, the Technical Program Chairman for the 1978 International Symposium on Circuits and Systems, the Vice President (1979) and President (1982) of the Circuits and Systems Society, and the Division I Director for 1984 and 1985. He was the recipient of an IEEE Centennial Medal, the 1985 Technical Achievement Award of the IEEE Acoustics, Speech, and Signal Processing Society, and the 1988 Education Award of the IEEE Circuits and Systems Society. In September 1988 he was named an Honorary Professor by the Institute of Acoustics and the Institute of Electronics of the Academia Sinica.



André Zaccarin (S'82–M'91) was born in Quebec City, Canada, in 1963. He received the B.Sc.A. and M.Sc. degrees in electrical engineering from Université Laval, Québec, Canada, in 1985 and 1987, respectively, and the M.A. and Ph.D. degrees in electrical engineering from Princeton University, Princeton, NJ, in 1988 and 1991, respectively.

Since June 1991, he has been an Assistant Professor with the Department of Electrical Engineering at Université Laval. His current research interests are in the areas of image processing, compression of still pictures and video sequences, and motion estimation for computer-vision applications.