Quadtree-Structured Variable-Size Block-Matching Motion Estimation with Minimal Error

Injong Rhee, Graham R. Martin, S. Muthukrishnan, and Roger A. Packwood

***Abstract*—This paper reports two efficient quadtree-based algorithms for variable-size block matching (VSBM) motion estimation. The schemes allow the dimensions of blocks to adapt to local activity within the image, and the total number of blocks in any frame can be varied while still accurately representing true motion. This permits adaptive bit allocation between the representation of displacement and residual data, and also the variation of the overall bit-rate on a frame-by-frame basis. The first algorithm computes the optimal selection of variable-sized blocks to provide the best-achievable prediction error under the fixed number of blocks for a quadtree-based VSBM technique. The algorithm employs an efficient dynamic programming technique utilizing the special structure of a quadtree. Although this algorithm is computationally intensive, it does provide a yardstick by which the performance of other more practical VSBM techniques can be measured. The second algorithm adopts a heuristic way to select variable-sized square blocks. It relies more on local motion information than on global error optimization. Experiments suggest that the effective use of local information contributes to minimizing the overall error. The result is a more computationally efficient VSBM technique than the optimal algorithm, but with a comparable prediction error.**

***Index Terms*—Dynamic programming, motion estimation, quadtree, variable-size block matching, video compression.**

I. Introduction

Many interframe coding schemes for video conferencing and multimedia employ motion-compensation techniques to exploit the temporal redundancy in an image sequence. Block-matching motion estimation, originally proposed by Jain and Jain [8], provides a simple and elegant way to identify and express motion, and hence, is commonly adopted in many video compression standards (e.g. ITU-T H.261/H.263, and MPEG-1, 2).

Each image frame is divided into a fixed number of nonoverlapping square blocks. For each block in the frame, a search is made in an earlier frame of the sequence over a predefined area of the image. The search is for the best matching block, the position which minimizes a distortion measure between the two sets of pixels comprising the blocks. Usually the criterion is to minimize either the mean square error (MSE) of the corresponding pixels, or the mean absolute error (MAE), which is easier to compute. The relative displacement between the two blocks is taken to be the *motion vector*. Typical block sizes are of the order of 16×16 pixels, and the maximum displacement might be ±64 pixels from the block's original position. Several search strategies are possible, usually using some kind of sampling mechanism, but the most straightforward approach is exhaustive search. This is computationally demanding but algorithmically simple, and relatively easily implemented in hardware.

The output of the motion-estimation algorithm comprises the motion vector for each block, and the pixel value differences between the blocks in the current frame and the "matched" blocks in the reference frame. We call this difference signal the *motion compensation error*, or simply *block error*. In many codes, the block error is transformed (e.g., discrete cosine transform) and the transform coefficients quantized in order to maintain an acceptable bit-rate, or compression ratio. For fixed bit-rates, large block errors result in increased information loss at this stage, and produce lower image quality. Thus, minimizing block error is an important goal of motion estimation.

Ideally, to achieve good video compression ratio and image quality, both the number of blocks and the block error have to be minimized because the motion vectors are encoded along with the block error. Unfortunately, this is a conflicting requirement, particularly with *fixed-size block matching* (FSBM), where the size of all the blocks is the same. In FSBM, increasing the block size is the only way to reduce the number of motion vectors. The success of this scheme relies on each block representing an area of single uniform motion, but as the block size is increased to reduce the number of motion vectors, this becomes increasingly unlikely, so a good match cannot be found.

Varying block sizes over different regions of an image based on the actual motion present in the regions would clearly provide a better optimization for this requirement. In *variable-size block matching* (VSBM) [2], [9], smaller blocks can be used to describe complex motion while larger blocks can be used in areas where the image content is stationary or undergoing uniform motion. However, its success depends on an appropriate selection of blocks. This poses an interesting optimization problem. Given two image frames f_1 and f_2, we wish to find B blocks that cover the entire frame f_2 and also minimize the total block error between f_1 and f_2. It is assumed that B is chosen to enable a target bit-rate to be obtained. Unfortunately, we do not have an efficient solution for this problem and conjecture that the problem is so hard (possibly NP-hard) that its solution is impractical. This is because arbitrary configurations of, possibly overlapping, blocks have to be considered.

Instead, we tackle a more practical problem, considering only those blocks possible by a *covering quadtree* decomposition. A covering quadtree is a quadtree where each node has four

Manuscript received May 7, 1997; revised June 5, 1999. This work was supported by the U.K. Engineering and Physical Sciences Research Council under Grant GR/J79997. This paper was recommended by Associate Editor H. Gharavi.

I. Rhee is with the Department of Computer Science, North Carolina State University, Raleigh, NC 27695-7534 USA.

G. R. Martin and R. A. Packwood are with the Department of Computer Science, University of Warwick, Coventry CV4 7AL U.K.

S. Muthukrishnan is with the Information Sciences Center, Bell Laboratories, Lucent Technologies, Murray Hill, NJ 07974 USA.

Publisher Item Identifier S 1051-8215(00)01632-3.

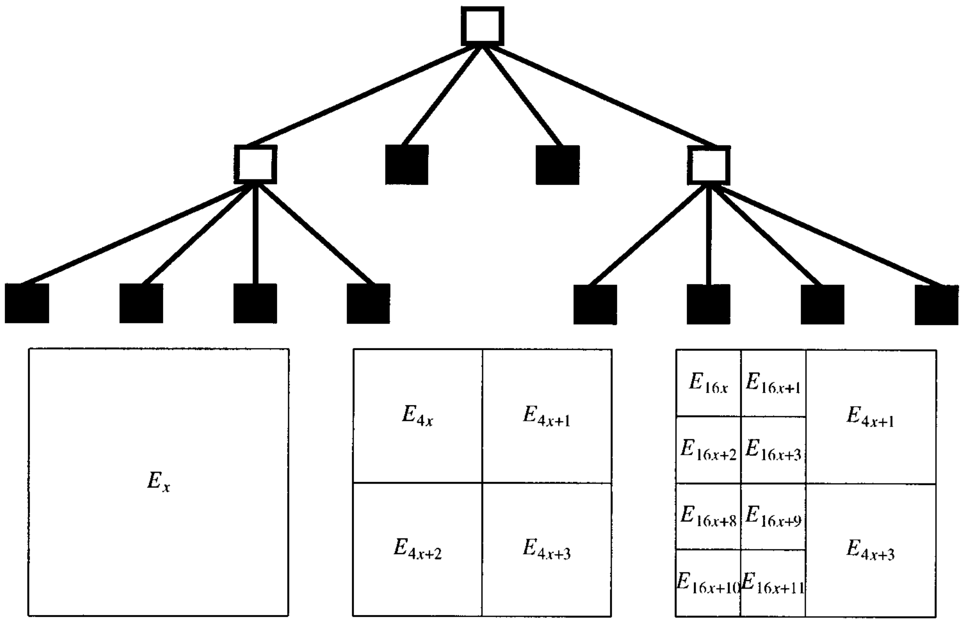

Fig. 1. A simple covering quadtree.

children or none with: 1) each node in the tree corresponding to a square block in the image; 2) four child blocks (i.e., nodes) of a block to the four equal-sized nonoverlapping square sub-blocks covering the block; and 3) the root node to the entire frame.[1] The resulting blocks correspond to the leaves of a covering tree. An example of a covering quadtree is shown in Fig. 1. The use of quadtrees is ubiquitous in image coding [19], [17], [12], [2], [9] because of the simplicity and efficiency in coding the block configuration, requiring only 1 bit per node of the tree.

In this letter, we present an efficient algorithm that finds the optimal covering quadtree of B leaves that gives the minimum total block error. The algorithm employs an efficient dynamic programming technique utilizing the special structure of a quadtree. The time complexity of the algorithms is $O(n^2B)$, where n^2 is the total number of pixels in a frame, which compares more favorably with the time complexity of the exhaustive search FSBM algorithm $(O(n^2))$ than other dynamic programming-optimal algorithms [13] whose time complexities are more than $O(n^4)$.

Minimizing distortion for a given block number does not necessarily yield the same result as minimizing distortion under a given bit budget. The latter requires computing the actual bit requirement of every candidate block through quantization and entropy coding. For real-time video coding, this process would be too time consuming. The overall bit rate can be approximated by the number of blocks if the bit usage of each coded block is similar (i.e., the bit rate is linear with the number of blocks). In VSBM, this assumption seems to hold, as smaller blocks tend to cover the areas with more distortion.

The optimal algorithm may not be suitable for a real-time code owing to its high computational demand. However, the algorithm provides a yardstick by which the performance of other practical VSBM techniques could be measured. This is because it gives the minimum compensation error among all quadtree algorithms that solve the minimization problem under a constraint based on the number of blocks.

[1]For convenience, we assume a square frame with a width of a power of two.

The high computational demand of the optimal algorithm provides the motivation to look for more computationally efficient VSBM algorithms exhibiting a comparable error. In this letter, we present one such heuristic VSBM algorithm that gives an error approximately the same as the optimal algorithm, but with much less computation. The algorithm relies more on local motion information than on global error optimization. It starts by computing sets of "candidate" motion vectors for fixed-size small blocks. Sibling blocks are then merged in a quadtree manner if their sets of candidate vectors contain at least one common vector. This algorithm is based on an observation that if neighboring regions undergo the same uniform motion, they must have at least one motion vector in common. Our experiments suggest that the effective use of local information contributes to minimizing the overall error. The computation overhead of the heuristic algorithm is also little more than that of the FSBM algorithm because most of the block matching is performed at the lowest level of the tree and the merging process can be implemented using efficient machine bitwise AND operations.

The performance of the various techniques was evaluated on an arbitrary 28 successive frames of the two image sequences, "Split Screen" and "Miss America," and on 30 frames of "Foreman," one of the Class B MPEG-4 video test sequences. An overall figure of merit indicates that, on average, the new heuristic VSBM algorithm differs from the optimal by only 1.69, whereas the FSBM gives 15.44 difference from the optimal.

Section II describes some related work on variable block-matching motion estimation, and Sections III and IV present the optimal algorithm and the heuristic VSBM scheme, respectively. Finally, the paper is summarized in Section IV, and some open problems are discussed.

II. Related Work

Puri *et al.* [15], Chan *et al.* [2], and Jelveh and Nandi [9] proposed using variable-sized blocks for motion estimation so that, where appropriate, large areas of uniform motion could be represented by relatively few blocks, thus minimizing the required number of motion vectors. They use a "top-down" approach in which initially large blocks are matched, and if for the best match of any block, the resulting error is above a prescribed threshold, then that block is split into four smaller blocks. This process is repeated until the maximum number of blocks, or locally minimum errors, are obtained. Finally, a process of re-merging small blocks to form large blocks is performed to remove blocks that do not contribute to improving image quality. They reported a significant improvement in quality over FSBM techniques for relatively low bit-rate coding.

Sullivan and Baker [18] used a variant of a Lagrange multiplier scheme that was previously applied to optimal bit-allocation in signal quantization [16]. A Lagrange multiplier, which is a fixed slope on a rate-distortion curve, is used to find the optimal tree minimizing distortion under a given bit rate. However, their algorithm can build the optimal tree *only* if the rate-distortion curve is convex. It is known that this convexity assumption is not generally realistic in signal coding [16]. In contrast, our optimal algorithm makes no assumption about a rate-distortion curve.

Bi and Chan [1] applied a tree-structured bit-allocation algorithm by Chou *et al.* [3], called a *generalized BFOS* (G-BFOS) algorithm, to motion estimation. The algorithm generates the trees that have rate and distortion points only on the convex hull of the rate-distortion curve. In the algorithm, a large number of pruned trees that generate points above the convex hull of the rate-distortion curve are ignored. Kiang *et al.* [10] proposed another approximation algorithm that improves the G-BFOS algorithm by removing in each step only those tree nodes with a common parent and no descendants. This technique identifies additional points which are inbetween two adjacent points on the convex hull.

For VSBM, several techniques have been proposed to reduce the number of encoded motion vectors and improve image fidelity [11], [21]. Kim and Lee [11] proposed a hierarchical motion-vector encoding scheme, in which only the vectors significantly different from those of their upper-level blocks are encoded. The motion vectors of the upper-level blocks are encoded as well. In general, a majority of the lower-level motion vectors are similar to those of the upper-level blocks. Zhang *et al.* [21] also developed a scheme that allows a block to have more than one motion vector, motion boundaries within a block being expressed by straight lines. This technique is applied to VSBM along with a similar motion vector reduction scheme as in [11]. Note that both Kim's and Zhang's schemes can be applied to any VSBM motion-estimation technique to improve bit rate and image quality.

III. Optimal VSBM Motion Estimation

In this section, we describe an efficient algorithm that finds the optimal covering quadtree of B variable-sized blocks which results in the minimum motion compensation error. In terms of motion estimation accuracy, it provides the best achievable performance of a quad-tree structured VSBM technique for any prescribed number of blocks.

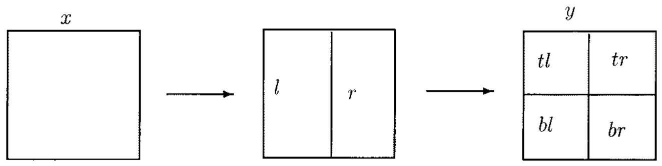

Fig. 2. Partitioning a node for recursive definition. Leftmost square: height = i + 1. Rightmost square: Height = i.

Assume the size of the input image array is $n \times n$. Without loss of generality, we assume n is a power of 2. Suppose that a covering quadtree has been superposed on top of this array. The root of this quadtree represents the entire array; its four children represent the four quadrants of the array, and each quadrant is recursively subdivided in the same manner and represented by corresponding children at successively deeper levels of the quadtree. Our goal is to identify the subtree with B leaves in this quadtree, such that the partitioning of the array induced by this tree gives the minimum MSE among all subtrees with B leaves. Note that each node x in a quadtree represents a $2^\ell \times 2^\ell$ square for some integer ℓ. Each such node has two disjoint $2^\ell \times 2^{\ell-1}$ sized left- and right-half rectangles, denoted l and r, respectively. Each rectangle of that form has two disjoint $2^{\ell-1} \times 2^{\ell-1}$ top and bottom squares, denoted t and b, respectively. Thus, each square x of size $2^\ell \times 2^\ell$ has four disjoint $2^{\ell-1} \times 2^{\ell-1}$ squares, which are denoted tl, tr, bl, and br in clockwise order starting with the top-left quadrant of x (see Fig. 2). The *height* of the leaf nodes is 0, their parents' height is 1, and so on.

Consider any node x. Define $E_x(i)$ to be the minimum (total) error if the subarray corresponding to x is partitioned into i blocks as per the quadtree partitioning. Then, we have

$$E_x(k) = \min_{i+j=k;1\le i,j} \left\{E_x^l(i) + E_x^r(j)\right\}. \tag{1}$$

Here, $E_x^l(i)$ is the minimum error in the left rectangle of the square at x with i blocks, and likewise for E_x^r with the right rectangle. We now define these two quantities recursively

$$E_x^l(k) = \min_{i+j=k;0\le i,j} \left\{E_x^{\mathrm{tl}}(i) + E_x^{\mathrm{bl}}(j)\right\} \tag{2}$$

$$E_x^r(k) = \min_{i+j=k;0\le i,j} \left\{E_x^{\mathrm{tr}}(i) + E_x^{\mathrm{br}}(j)\right\}. \tag{3}$$

Here, $E^{\mathrm{tl}}(i)$ is the minimum error in the top square of the rectangle l with i blocks, and likewise for E^{bl} with the bottom square of the left rectangle. A similar recursive definition holds for the right rectangle and that defines E^{tr} and E^{br}. Now we observe merely that $E_x^{\mathrm{tl}}, E_x^{\mathrm{tr}}, E_x^{\mathrm{br}}, E_x^{\mathrm{bl}}$ are $E_{4x}, E_{4x+1}, E_{4x+2}$, and E_{4x+3}, in that order. Finally, $E_x(1)$ is simply the best motion vector for that block x with respect to the previous image (more on this below). That completes the recursive description of E_x. Clearly $E_x(B)$, where x is the root of the quadtree gives the minimum error possible with B blocks.

Direct evaluation of the recursive definition above will require exponential time, but dynamic programming [7] applies here, and each of the $E(\,)$ values can be computed using others computed at lower levels in a bottom-up fashion. That will give an efficient algorithm based on this recursive definition. Note that it is straightforward to modify such a dynamic program to actually determine the quadtree which induces the partition with minimum MSE. That completes the description of our entire algorithm.

It remains for us to calculate the running time of our algorithm. This has two parts.

1) The time to calculate $E_x(1)$ for each node x of the quadtree.
2) Given $E_x(1)$, calculate $E_x(i)$ for each value of $i, 2 \le i \le B$, for each node x of the quadtree.

First we consider 2) above. Assume $E_x(j), 1 \le j \le B$ has been computed for any node x at height at most $i-1$ in the quadtree. Consider the nodes at height i. There are $(n/2^i) \times (n/2^i)$ nodes, each representing a disjoint block of size $2^i \times 2^i$. First, we compute $E_x^l(k)$ and $E_x^r(k)$ for all nodes x at height i and for all $k, 1 \le k \le B$. Computing $E_x^l(k)$ for a given k takes time $O(k)$ using (2), and likewise for $E_x^r(k)$ using (3). Over all k, this takes time $\sum_{1 \le k \le B} O(k) = O(B^2)$ for a given x. Next, we calculate $E_x(k)$ for all $1 \le k \le B$ for node x at height i. Since that takes $O(k)$ time by (1) for each $E_x(k)$, the total time taken for x is $\sum_{1 \le k \le B} O(k) = O(B^2)$, and the total time for all nodes at height i is $(n/2^i) \times (n/2^i) \times O(B^2)$. Finally, repeating this procedure at each level of the quadtree, the total time taken is

$$\sum_{i \ge 0} \frac{n}{2^i} \times \frac{n}{2^i} \times O(B^2) = O(B^2)\left(n^2 + \frac{n^2}{4} + \cdots\right) = O(n^2B^2).$$

Thus, the total time for 2) is $O(n^2B^2)$. A more detailed enumeration gives an improved analysis as shown below.

Case 1: Consider any height i where $4^i \le B$. At this height, there are $n/2^i \times n/2^i$ blocks, each of size $2^i \times 2^i$. For each such block x, we need compute only 4^i values, namely $E_x(k), 1 \le k \le 4^i$, since $E_x(k) = E_x(4^i)$ for all $B \ge k \ge 4^i$. (Informally, we cannot divide a 4^i-sized block into more than 4^i blocks any better than to divide it into at most 4^i blocks). Thus, the total time taken for computing all the relevant values for the nodes at height less than or equal to i, for $4^i \le B$, is

$$\sum_{0 \le i \le \log_4 B} \frac{n}{2^i} \times \frac{n}{2^i} \times O((4^i)^2)$$

which equals

$$\sum_{0 \le i \le \log_4 B} n^2 \times O(4^i) = O(n^2B).$$

The total time for this part is thus $O(n^2B)$.

Case 2: Consider nodes at height i where $4^i > B$. Using the basic argument above, the time taken for this part is

$$\sum_{i \ge \log_4 B} \frac{n}{2^i} \times \frac{n}{2^i} \times O(B^2) = O(B^2)\left(\frac{n^2}{B} + \frac{n^2}{4B} + \cdots\right) = O(n^2B).$$

We can conclude that this case, too, takes time $O(n^2B)$.

Summing both parts, the total time is $O(n^2B)$.

Now we turn to the task 1) above; namely, calculating $E_x(1)$ for all nodes x in the quadtree. This is defined to be the minimum block error between x and a block of the same size as x with its top-left corner in a square symmetric region of $c \times c$ around the top-left corner of x in the previous frame. Typically c is taken to be 64 pixels, independent of n, and for that reason, we will consider $c = O(1)$ in this section.

The straightforward way to compute $E_x(1)$ is as follows. At height i in the quadtree, we have $n/2^i \times n/2^i = n^2/4^i$ blocks, each of size $2^i \times 2^i = 4^i$. For each such block, we can compare it against each of the c^2 positions for placing its top-left corner, and can explicitly compute the MSE for each such placement and retain the minimum. This takes time $c^2 4^i$ for each block, and $c^2 4^i n^2/4^i = O(c^2n^2)$ time in all. Over all possible heights, this takes time $O(c^2n^2 \log n) = O(n^2 \log n)$ since we assumed $c = O(1)$.

We can compute $E_x(1)$'s more efficiently in a bottom-up manner as follows. Say $E_x(1)$ has been computed for all nodes x at height of, at most, $i-1$. We maintain, for each block in the current frame, its MSE, with each block in the previous frame with its top-left corner in one of the c^2 possible positions. Consider any node x at height i. For each potential top-left corner of x in the previous frame, there is only one possible position for the top-left corner for each of its four quadrant squares. But, for each such placement, we have already calculated the MSE of that quadrant of x and its corresponding square in the previous frame at height $i-1$. Hence, in $O(1)$ time, we can consult all four quadrants and determine the MSE for x. Thus, the time taken for each node x is only $O(c^2)$. Therefore, the time taken for all the nodes at height i is $O(n^2/4^ic^2)$, which over the entire quadtree becomes $O(n^2c^2) = O(n^2)$, since we have assumed $c = O(1)$. The space used is $O(n^2c^2) = O(n^2)$ as well. That gives the following theorem.

Theorem 3.1: There exists an algorithm that finds the quadtree of, at most, B leaves, minimizing the MSE in time $O(n^2B)$ and space $O(n^2)$.

We remark that our particular manner of solving this problem, namely, defining $E(\,)$ for a square in terms of that for its constituent halves (rectangles) which in turn is defined in terms of that for smaller squares (to complete the recursion), is particularly efficient. An alternative and more straightforward way to solve this problem would be to write the $E(\,)$ for a block in terms of that of its four quadrants. That would lead to an algorithm taking time $\Omega(n^2B^3)$ which is prohibitive since B may be large.

IV. Heuristic VSBM Motion Estimation

In this section, we describe a heuristic technique that uses only a fraction of the computational time of the optimal algorithm, but gives a near-optimal selection of blocks.

Our heuristic algorithm is based on the following observation. When an object moves, the motion perceived in a local window can be ambiguous, meaning that it is not possible to determine the true motion using only local information. For example, in Fig. 3, as the object moves to the top-right corner, both of the motion vectors shown are equally probable candidates of the true motion within that window. This problem is called the

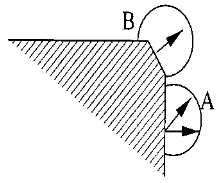

Fig. 3. Aperture problem.

aperture problem [14]. The true motion can be recovered by incorporating more global information. In Fig. 3, the vector shown in window B represents the true motion as the motion is perpendicular to the object. By choosing the common "candidate" motion vectors in windows A and B, the true motion of the moving object can be recovered.

Based on this observation, our technique first computes sets of "candidate" motion vectors for fixed-size small blocks, and neighboring blocks are then merged in a quadtree manner if they have at least one vector in common. Given an image sequence $f_1, f_2, \cdots$, the algorithm first divides the image frame f_i into small fixed-size blocks (we used 4 $\times$ 4 blocks in our experiments). Motion information for each of these small blocks is obtained by block matching. We denote each block by a tuple (x, y, s) where (x, y) is the coordinate of the upper-leftmost pixel of the block and s is the length of a side of the square block. Given a block (x, y, s) and a predefined search window of motion vectors V, we calculate the MSE (or mean absolute error) between blocks (x, y, s) in f_i and $(x + x', y + y', s)$ in f_{i-1} for each (x', y') in V. For each block b in f_i, we obtain a set of motion vectors I_b, called the *initial set* of b, whose matching error is less than a prescribed threshold. The initial set of vectors can be obtained using the exhaustive search technique or more efficient subsampled search strategies (e.g. [20]).

Let us define a set of motion vectors IS_b for each block b in the tree, called the *intersection set* of b as follows. If b is a leaf, $IS_b = I_b$. Otherwise, $IS_b = IS_i \cap IS_j \cap IS_k \cap IS_l$ where i, j, k, and l are the children of b. We merge four sibling blocks into a parent block if and only if the intersection set of each sibling block is not empty and the intersection set of the parent block is not empty. The reasoning behind this process is as follows. When IS_b is empty, there is no vector that is common to the four children of block b. This means that there exists at least one child/descendent block b' that has moved differently from the other sub-blocks of block b. Merging those blocks into b will provide an incorrect motion vector for b'. We repeat this process at each level of the tree, from the bottom level to the top. The motion vector for a block which cannot be merged into its parent or for a block which does not have any candidate motion vector (i.e., $I_b = \emptyset$) is chosen to be the one associated with the minimum error of those within its intersection set.

The merging process is illustrated in Fig. 4, which shows the initial sets of motion vectors for 16 blocks.

Each of blocks 11, 12, 15, and 16 has one common motion vector, so the intersection set of their parent block has one member. Consequently the four blocks are merged and the resulting motion vector is that contained in the intersection set. The intersection set of the parent of blocks 1, 2, 5, and 6 has two members. Again, the blocks are merged, and the motion

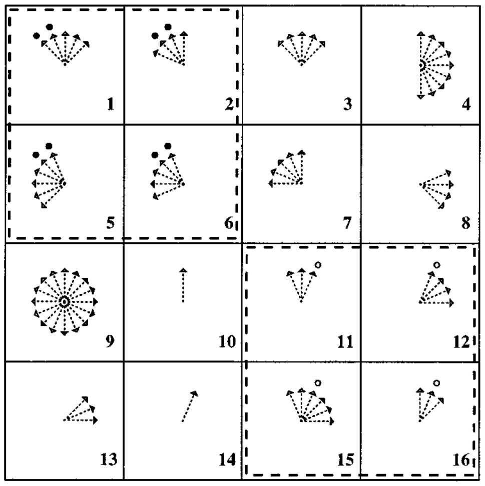

Fig. 4. Initial sets of motion vectors and merging of blocks.

vector is chosen to be the one with the smallest block error of the two candidates in the merged block. This is simply found by summing the residual error of the four blocks for each of the two vectors. The remaining blocks in the example cannot be merged as the intersection sets of their parent blocks are empty. The motion vector for each nonmerged child is selected as the one associated with the minimum error.

This bottom-up algorithm generally produces better prediction than the top-down algorithms. The top-down VSBM methods [15], [2], [9] suffer from the 'majority effect'. In top-down 'match or split' algorithms, a large block may be considered matched when having an acceptable block error, but a small area of the block may represent a feature with disparate motion. Essentially this is ignored because the error is biased by the majority of the pixels. This effect does not occur in our bottom-up scheme in which regions with disparate motion would not be merged with other areas. Bottom-up VSBM techniques should better represent the true motion within the image.

The technique can be efficiently implemented using machine-level bitwise intersection operations. The initial and intersection sets (I_b, IS_b) can be represented by simple bit vectors, a set corresponding to a 15 $\times$ 15 search area having 225 possible vectors can be stored in an array of 8 ($\times$ 32 bit) integers. Intersection of these sets is trivial using a logical AND operation.

The threshold, which determines which vectors are included in the initial sets, is calculated on a frame by frame basis. Essentially, the threshold controls the number of blocks produced in the quad-tree, and this allows the error performance of VSBM to be compared with other techniques in a meaningful way. For the experimental results detailed in the next section, the threshold was determined using an iterative technique. However, it is found that the required threshold is proportional to the minimum mean absolute matched error of the entire frame, and in a practical codec it could be calculated once block matching has been performed using the initial (4 $\times$ 4) blocks. In concept, this is similar to the method employed by Jelveh and Nandi [9], which allows the threshold to adapt to the degree of motion in the image.

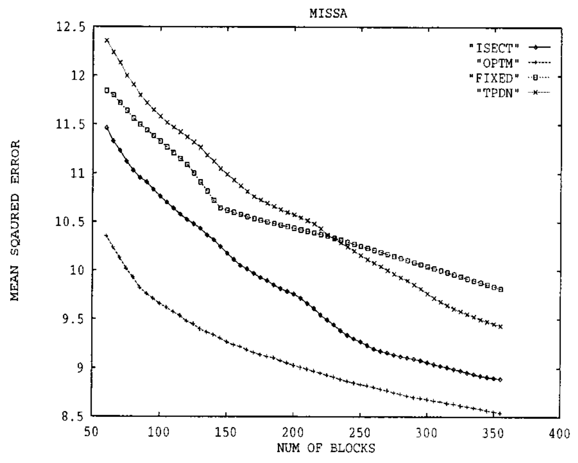

Fig. 5. Comparative results over 28 frames of "Miss America."

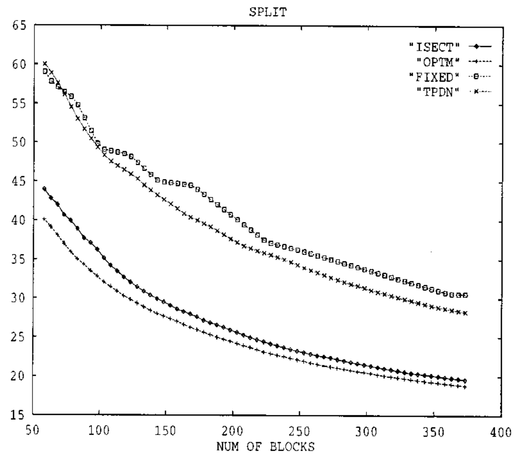

Fig. 6. Comparative results over 28 frames of "Split Screen."

The computational cost of this VSBM technique is little more than that of FSBM. In fact, block matching is performed only at the bottom level, and then the "minimal error" tree is formed using a block merging process. This means that, computationally, the technique is not only significantly less demanding than the optimal VSBM algorithm, but also more efficient than other VSBM methods where block matching has be to performed each time a block is split or merged. Chan *et al.* [2], for instance, estimated that from a large number of simulations on various image sequences, their technique was about three times as computationally demanding as FSBM.

V. Performance Evaluation

We evaluate the performance using three image sequences: "Miss America," "Split Screen," and "Foreman." Each of the sequences "Miss America" and "Split Screen" comprise 28 frames, each of 256 $\times$ 256 pixels. "Foreman" is one of the (Class B) MPEG-4 video test sequences, made available in ITU-R 601 format. We converted the sequence to CIF format (a procedure commonly adopted in low bit rate codecs) and then processed only a centralized window of 256 $\times$ 256 pixels, to achieve an image size compatible with the other two sequences. To further reduce the computational requirement of the tests, only 30 of the 300 available frames of the "Foreman" sequence were used. Results are presented for FSBM ("fixed"), top-down VSBM ("tpdn") of Chan *et al.* [2], the new bottom-up VSBM technique ("isect") and the optimal VSBM method ("optm"). An exhaustive search was adopted for all the tested techniques.

The performance of each method is compared in terms of the MSE, a measure of the distortion introduced in the prediction process. This was considered to be most appropriate as the overall objective is to find a block matching motion estimation method which provides the highest fidelity. Fig. 5 shows the MSE over different numbers of blocks for the "Miss America" sequence. The MSE is given as the average value over 28 frames, which is considered more representative than presenting results for one (typical) frame. The range of block numbers presented is chosen to be appropriate for a low bit-rate code (e.g,. ITU-T H.263) operating at 64-kbits/s with approximately CIF-format data.

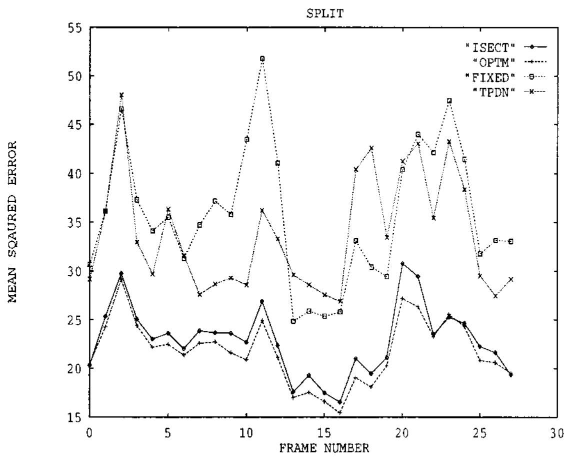

Fig. 7. Comparative results for 256 blocks/frame of "Split Screen."

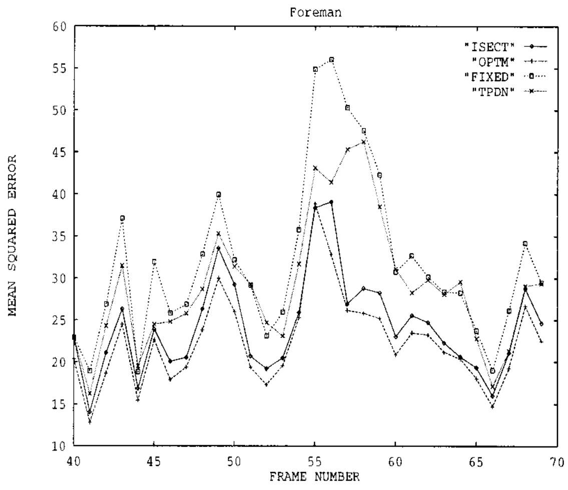

Fig. 8. Comparative results for 256 blocks/frame of "Foreman."

As expected, the optimal method shows the best performance, and the bottom-up technique is clearly better than both FSBM and top-down VSBM. However, the advantage is only marginal because the "Miss America" sequence contains little motion, and all of the techniques work well.

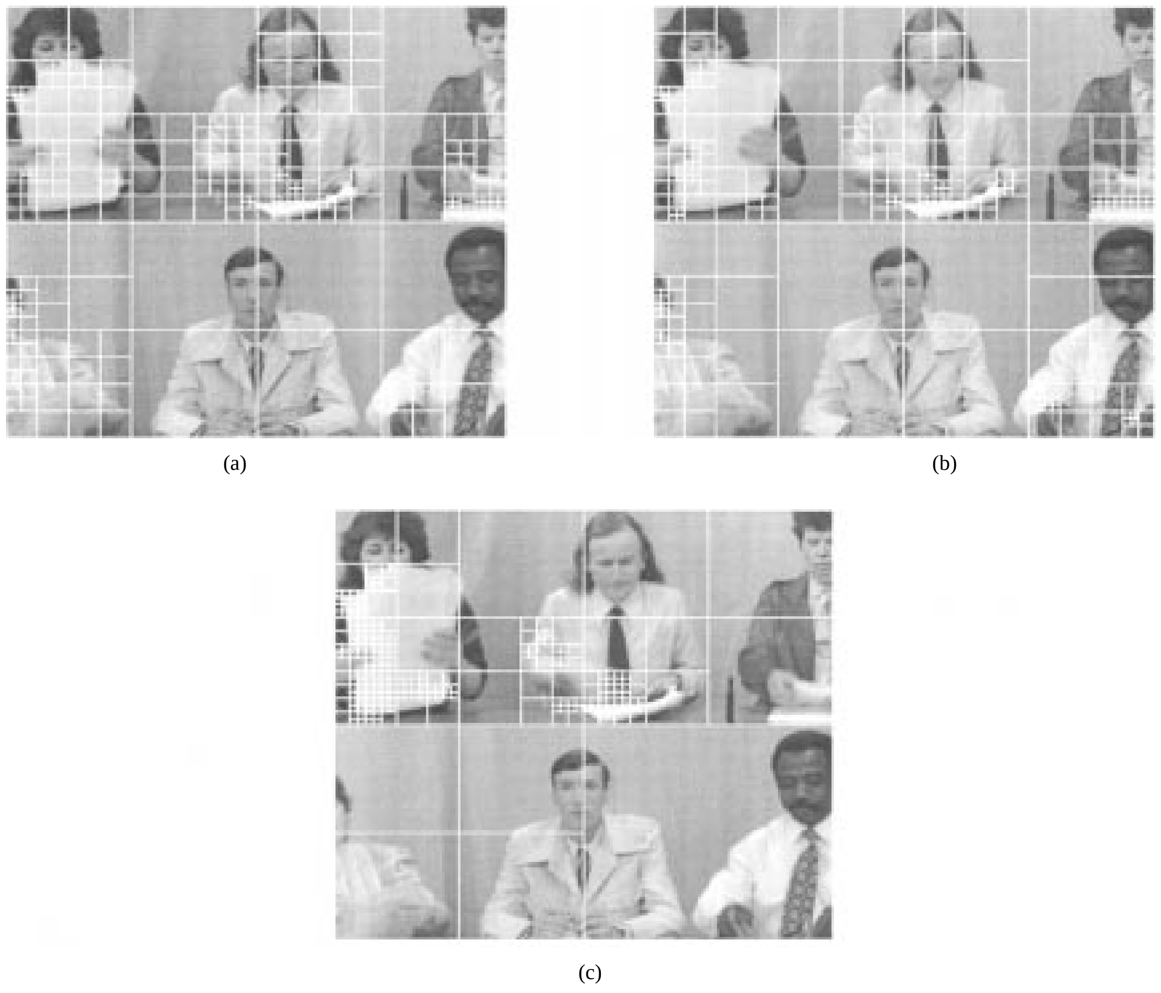

Fig. 9. Block structures for each VSBM technique of "Split Screen" (256 blocks). (a) OPT. (b) ISECT. (c) TPDN.

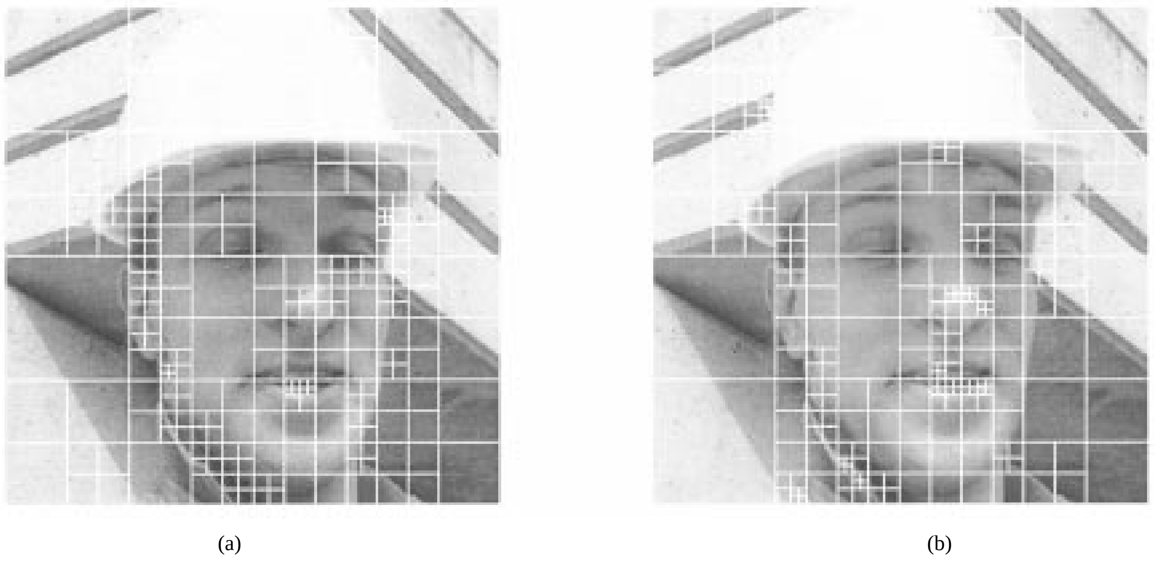

Fig. 10. Block structures for optimal and bottom-up VSBM techniques of "Foreman" (232 blocks). (a) OPT. (b) ISECT.

Averaged MSE results for the "Split Screen" sequence are presented in Fig. 6. It is noticeable that the averaged errors are much larger than for the "Miss America" sequence. In the "Split Screen" sequence, many small regions undergo different translational motion, providing a more rigorous test for the various techniques. Again, the optimal VSBM method shows the best performance, but the new bottom-up VSBM technique performs extremely well, especially when compared with FSBM and top-down VSBM.

Figs. 5 and 6 could hide aberrant behavior if, for a prescribed number of blocks, there were large differences in MSE from frame to frame. However, this is not the case. Fig. 7 shows the frame-to-frame variation in MSE for the "Split Screen" sequence using a fixed number of blocks (256) in each frame. The MSE does vary by a factor of two, above and below the mean level, due to changes in the degree of motion between frames, but the techniques approximately track each other through the sequence with no anomalous behavior. Similar results are shown in Fig. 8 for frames 40–70 of the "Foreman" sequence. For both sequences, the "'bottom-up" VSBM technique provides near-optimal results.

Fig. 9 shows the same frame (from the "Split Screen" sequence) which has been motion compensated using the optimal VSBM (OPT), bottom-up VSBM (ISECT) techniques, respectively. The variable-sized block structure is superimposed on each image. It is apparent that the new bottom-up VSBM technique has made very similar decisions to the optimal VSBM method, whereas the top-down VSBM technique has chosen a very different block structure, resulting in an increased error.

Fig. 10 shows the optimal (OPT) and bottom-up (ISECT) SBM block structures for a frame of "Foreman." Very similar decisions to merge blocks have been made. The stationary background on the periphery of the image has been represented by comparatively few blocks, whereas the complex movements of the mouth, nose, and chin are represented by much smaller blocks, and hence, a larger number of motion vectors.

To generate overall figures of merit, using the optimal VSBM method as a baseline, the MSE differences between each technique and the optimal were averaged across the complete frame sequence and for all block numbers. For the "Split Screen" sequence, the new bottom-up VSBM technique showed a difference of only 1.69, whereas for FSBM and top-down VSBM, the figures were 15.44 and 13.32, respectively.

VI. Conclusion

In this letter, we investigated VSBM motion-estimation techniques. They demonstrate considerable advantages over FSBM methods which rely on each block representing an area of uniform translational motion. With VSBM, the size of each block adapts to local activity within the image, larger blocks being used in large areas of stationary background or uniform motion, and smaller blocks where the movement is localized or complex. VSBM also allows the total number of blocks in any frame to be varied while still representing true motion fairly accurately.

We described an algorithm based on a quadtree structure which results in the optimal selection of variable-sized blocks, and thus, the minimum total error. Although it is computationally demanding, and hence, impractical for real-time codes, it does provide a yardstick by which the performance of other VSBM techniques can be measured. The method is based on an exhaustive tree search, and provides the best-achievable results for a quadtree-based VSBM scheme.

We also described a new bottom-up VSBM technique which is as computationally efficient as FSBM, and yet provides near-optimal results. Block matching is performed only once, using small square blocks. Blocks are then merged in a quadtree manner depending on whether they have candidate motion vectors in common. This bottom-up approach has a number of advantages over other known VSBM techniques.

The computational requirement of the bottom-up algorithm is minimal as the search for matching blocks is no more demanding than for FSBM. Following an evaluation of the various techniques using real image sequences, the new bottom-up VSBM method performs almost as well as the best possible quadtree-based scheme. The residual error produced is significantly lower than for FSBM and also better than for top-down VSBM techniques.

It would be of interest to see if other ways of partitioning an image (besides that based on quadtrees) will be more desirable. One such method is to divide the rows and columns into p intervals independently and consider the p^2 blocks induced by such a division. Another issue is whether "thinning" the optimal algorithm leads to significant loss. That is, instead of considering $E_x(i)$ for node x for each $1 \leq i \leq B$, we may only look at $E_x(2^j)$ for $0 \leq j \leq \log_2 B$. Note that such an algorithm takes only $O(n^2 \log B)$ time and, therefore, will be more efficient than the optimal algorithm.

Acknowledgment

The authors thank the anonymous referees for their critical reading of earlier versions of this paper and many helpful comments.

References

[1] H. Bi and W. Chan, "Rate-constrained hierarchical motion estimationusing BFOS tree pruning," in *Proc. ICASSP*, 1996, pp. 2315–2317.

[2] M. Chan, Y. Yu, and A. Constantinides, "Variable size block matching motion compensation with applications to video coding," *Proc. Inst. Elect. Eng.*, pt. I, vol. 137, no. 4, pp. 205–212, Aug. 1990.

[3] P. Chou, T. Lookabaugh, and R. Gray, "Optimal pruning with applications to tree-structured source coding and modeling," *IEEE Trans. Inform. Theory*, vol. 35, pp. 299–315, Mar. 1989.

[4] *Video Codec for Audiovisual Services at p^{*64} kbit/s*, 1990. ITU-T (CCITT) Recommendation H.261, 1990.

[5] *Video Coding for Low Bit Rate Communication*. ITU-T Recommendation H.263, 1996.

[6] *Information Technology—Coding of Moving Pictures and Associated Audio for Digital Storage Media up to about 1.5 Mbit/s*. ISO/IEC IS 11172, 1993.

[7] T. Cormen, C. Leiserson, and R. Rivest, *Introduction to Algorithms*. Cambridge, MA: MIT Press, 1990.

[8] J. R. Jain and A. K. Jain, "Displacement measurement and its application in interframe image coding," *IEEE Trans. Commun.*, vol. COM-29, pp. 1799–1808, Dec. 1981.

[9] H. Jelveh and A. Nandi, "Improved variable size block matching motion compensation for video conferencing applications," in *Digital Signal Processing*, A. Cappellini and A. Constantinides, Eds. Berlin, Germany: Springer-Verlag, 1991, pp. 391–396.

[10] S. Kiang, R. Baker, G. Sullivan, and C. Chiu, "Recursive optimal pruning with applications to tree structured vector quantizers," *IEEE Trans. Image Processing*, vol. 1, pp. 162–169, Apr. 1992.

[11] J. Kim and S. Lee, "Hierarchical variable block size motion estimation technique for motion sequence coding," *Opt. Eng.*, vol. 33, no. 8, pp. 2553–2561, 1994.

[12] J. Lee, "Optimal quadtree for variable block size motion estimation," in *Proc. IEEE Int. Conf. Image Processing (ICIP'95)*, Washington, DC, pp. 480–483.

[13] G. Martin, R. Packwood, and I. Rhee, “Variable size block matching motion estimation with minimal error,” in *Proc. IS&T/SPIE Symp. Electronic Imaging: Science and Technology*, vol. 2668, San Jose, CA, Jan. 1996, pp. 324–333.

[14] D. Marr and S. Ullman, “Directional selectivity and its use in early visual processing,” *Proc. Royal Soc. London*, vol. 211.B, pp. 151–180, 1981.

[15] A. Puri, H. Hang, and D. Schilling, “Interframe coding with variable block size motion compensation,” in *Proc. GLOBECOM'87*, pp. 65–69.

[16] Y. Shoham and A. Gersho, “Efficient bit allocation for an arbitrary set of quantizers,” *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 36, pp. 1445–1453, Sept. 1988.

[17] P. Strobach, “Quadtree-structured recursive plane decomposition coding of images,” *IEEE Trans. Signal Processing*, vol. 39, pp. 1380–1397, June 1991.

[18] G. Sullivan and R. Baker, “Efficient quadtree coding of images and video,” *IEEE Trans. Image Processing*, vol. 3, pp. 327–331, 1994.

[19] D. Vaisey and A. Gersho, “Variable block-size image coding,” *IEEE Trans. Signal Processing*, vol. 40, pp. 2040–2060, Aug. 1992.

[20] A. Zaccarin and B. Liu, “Fast algorithms for block motion estimation,” *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, vol. III, pp. 449–452, Mar. 1992.

[21] K. Zhang, M. Z. Bober, and J. V. Kittler, “Variable block size video coding with motion prediction and motion segmentation,” in *Proc. SPIE Conf. Digital Video Compression: Algorithms and Technologies*, 1995, pp. 62–70.