Variable size block matching motion estimation with minimal error

Graham R. Martin, Roger A. Packwood, Injong Rhee

Department of Computer Science, University of Warwick
Coventry CV4 7AL, United Kingdom

ABSTRACT

We report two techniques for variable size block matching (VSBM) motion compensation. Firstly an algorithm is
described which, based on a quad-tree structure, results in the optimal selection of variable-sized square blocks. It is applied
in a VSBM scheme in which the total mean squared error (MSE) is minimized. This provides the best-achievable
performance for a quad-tree based VSBM technique. Although it is computationally demanding and hence impractical for
real-time codecs, it does provide a yardstick by which the performance of other VSBM techniques can be measured.
Secondly, a new VSBM algorithm which adopts a ‘bottom-up’ approach is described. The technique starts by computing sets
of ‘candidate’ motion vectors for fixed-size small blocks. Blocks are then effectively merged in a quad-tree manner if they
have similar motion vectors. The result is a computationally-efficient VSBM technique which attempts to estimate the ‘true’
motion within the image.

Both methods have been tested on a number of real image sequences. In all cases the new ‘bottom-up’ technique was only
marginally worse than the optimal VSBM method but significantly better than fixed-size block matching and other known
VSBM implementations.

Keywords: motion estimation, block matching, interframe coding

1. INTRODUCTION

Motion compensation is an important component of many low bit-rate video coding schemes. A prediction of the
current frame is obtained from one or more previous frames, and the prediction together with the error (difference) signal is
coded for subsequent storage or transmission. The fidelity of the decoded frame and the achieved data compression ratio are
dependent on the accuracy of the prediction and the information content of the error signal. In practice, a tradeoff is made
between the quality of the reconstructed sequence and the available bandwidth.

Currently, most videoconferencing codecs (eg. ITU-T H.261 1, H.263 2) and multimedia video codecs (eg. MPEG 3) employ
block matching algorithms of the type originally proposed by Jain and Jain 4. The image frame is divided into a fixed number
of small blocks, and for each block a search is conducted within a predefined window of the previous frame to find a best
match. The acceptance criterion is usually based on minimizing the mean square error or mean absolute error between the
two sets of pixels, and the relative displacement between the two blocks is taken to be the motion vector. The computational
requirements depend on the search method, the size of the search window and the acceptance criterion. The technique is
relatively simple to implement and thus widely adopted.

Ideally, to achieve maximum data compression, it is necessary to minimize both the number of motion vectors (number of
blocks) and the difference between matched blocks. These are conflicting requirements, particularly with ‘fixed size block
matching’ (FSBM). The success of the scheme relies on each block of pixels representing an area of uniform motion, but as
the block size is increased to reduce the number of motion vectors it becomes increasingly unlikely that a good match can be
found. This deficiency has been addressed by a number of researchers. Chan, Yu and Constantinides 5 proposed using
variable sized blocks 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 used a ‘top-down’ approach in which, initially, comparatively
large blocks are matched. 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, which are then matched independently. This process is repeated until the maximum number
of blocks, or locally-minimum errors, are obtained. Chan et al reported a significant success with the scheme, and for relatively low bitrates an improved quality was obtained when compared with FSBM.

It is without doubt that, in general, variable size block matching (VSBM) techniques should perform better than FSBM methods. However, their success is dependent on an appropriate selection of different size blocks to cover the entire image. Clearly, given any fixed number of blocks, in order to obtain the best frame prediction from a VSBM scheme it is necessary to select that set of variable-sized blocks which, when matched with similar sized blocks in the previous frame, results in the minimum total error. Computationally, this would be impractical to determine in a real-time codec implementation, but it would result in the optimal solution and provide a yardstick by which the performance of other VSBM techniques could be measured.

In this paper we describe an algorithm based on a quad-tree structure which results in the optimal selection of variable-sized square blocks. A tree structure is chosen for simplicity and because it requires a minimal number of bits to encode. It would be trivial to modify the algorithm for other types of tree, e.g., binary. The method provides an exhaustive tree search to generate the optimal segmentation in terms of the minimum residual error. The algorithm has been applied in a VSBM scheme in which the total mean absolute error (MAE) is minimized. It has been tested on a number of video test sequences and complete sets of results have been obtained for images which have been divided into different numbers of blocks. The performance, then, represents the best achievable using a quad-tree based VSBM technique. In order to conduct a comparison, the same test data was applied to Chan’s algorithm again for different numbers of blocks. In each trial, the total MSE was notably worse than that obtained with the optimal quad-tree algorithm, but of course the computational requirement was very much less. The performance comparison does indicate that there exists the potential for more computationally-efficient VSBM algorithms which exhibit a lower total error.

Motivated by the potential, we present a new VSBM technique which performs very well when compared with the optimal algorithm. The algorithm relies more on local motion information than on global error optimization in order to find the set of variable-sized blocks. Our experiments suggest that the effective use of local information contributes to minimizing the overall error.

The performance of the various techniques was evaluated on an arbitrary 28 successive frames of each 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 for the ‘Split Screen’ sequence, which contains significant movement, the MSE for the new (bottom-up) VSBM algorithm differs from the optimal by only 1.69 whereas fixed size block matching differs from the optimal by 15.44. For comparison, an implementation of ‘top-down’ VSBM showed an MSE difference of 13.32.

The optimal, exhaustive quad-tree search, VSBM algorithm is described in section 2. The new VSBM technique which provides near-optimal results at a fraction of the computational cost is presented in section 3. The results of comparing the two methods with fixed-size block matching are discussed in section 4, and the work is summarized in section 5.

2. OPTIMAL VSBM IMPLEMENTATION

In block matching motion compensation there is a direct relationship between the size of the block and the error (or difference) between the current block and the best match in the previous frame. As the block gets larger, the error is also likely to get larger because all the pixels in the block are unlikely to experience the same translational motion. Consequently, a single vector is then insufficient to describe the motion. An ideal VSBM technique should find the optimal tradeoff between the size of the blocks (and hence the number of blocks), and the total error associated with them.

In the ‘optimal’ VSBM algorithm, we impose a complete quad-tree on the block structure of a frame. We denote a square block by \((x, y, s)\) where \((x, y)\) are the coordinates of the upper left-most pixel of the block, and \(s\) is the length of a side of the block. The frame is initially divided into identical-sized small blocks of size \(s_{\text{min}}\). They constitute the leaves of the tree. A parent block \((x, y, s)\) is defined to be \((2s \left\lceil \frac{x}{2s} \right\rceil, 2s \left\lceil \frac{y}{2s} \right\rceil, 2s)\) and the parent node of the node corresponding to the block is
determined accordingly. We shall refer to the root of a tree as node 0, and the four children of a node \( x \) as \( 4x, 4x+1, 4x+2 \) and \( 4x+3 \). The output of block matching motion estimation is a set of non-overlapping blocks which, together, ‘cover’ the entire frame. We define this set of blocks to be a covering quad-tree; from now on simply referred to as a ‘tree’. The principle is illustrated in Figure 1.

![Diagram of a tree with labeled nodes and blocks]

Fig. 1. Decomposition and the resulting ‘covering quad-tree’.

Clearly, there are many tree structures, and we can easily see that any tree with height less than \( \log_4 n \), where \( n \) is the total number of blocks of size \( s_{\text{min}} \), can be mapped uniquely to a set of non-overlapping blocks which covers the entire input frame. The error of the tree, the total error of the matched blocks comprising the tree, is the error of the motion compensated frame. Given a required number of blocks \( B \) and two consecutive frames \( f_{i-1} \) and \( f_i \), the optimal block matching requirement is to find a tree with \( B \) leaves whose error is minimal among all possible trees with \( B \) leaves. We call such a tree the \( B \)-optimal tree of \( f_i \). The solution follows.

Let \( T_x(B) \) be the \( B \)-optimal tree whose root is \( x \) and which covers only the area of the block corresponding to node \( x \). Let \( E_x(B) \) be the error of \( T_x(B) \). Let \( \pi_B \) be the set of tuples \( (i, j, k, l) \) s.t. \( B = i + j + k + l \).

The following is true:

\[
E_x(B) = \min_{(i, j, k, l) \in \pi_B} \{ E_{4x}(i) + E_{4x+1}(j) + E_{4x+2}(k) + E_{4x+3}(l) \}
\]

where \( E_x(1) \) is the error of the block corresponding to node \( x \).

By solving this recursive equation, we can calculate the minimum error \( E_0(B) \) and hence obtain the \( B \)-optimal tree for the entire frame, \( T_0(B) \).

A naive solution of the recursive equation is to test all possible trees with \( B \) leaves. However, to avoid the high (exponential) computational requirement, we adopt a dynamic programming technique. The following pseudo-code illustrates our implementation.
for \((lev = 1; lev \leq \log_4 n; lev = lev + 1)\)
  
  for all nodes \(x\) at level \(lev\)
    
    for \((i = 1; i \leq 4^{lev-1}; i = i + 3)\)
    
    for \((j = 1; j \leq 4^{lev-1}; j = j + 3)\)
    
    for \((k = 1; k \leq 4^{lev-1}; k = k + 3)\)
    
    for \((l = 1; l \leq 4^{lev-1}; l = l + 3)\)
      
      \[E_x(i + j + k + l) = \min \{ E_x(i + j + k + l), E_{4x}(i) + E_{4x+1}(j) + E_{4x+2}(k) + E_{4x+3}(l) \}\]

The computational cost of the above implementation is \(O(n^2 \log(n))\), excluding the cost of computing the motion vectors for each block corresponding to each node in the initial complete quad-tree.

The performance of the Optimal VSBM algorithm is considered in section 4.

### 3. A NEAR-OPTIMAL VSBM ALGORITHM

In the ‘near-optimal’ (bottom-up) VSBM algorithm proposed, we first divide 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 gathered by block matching. Given a block \((x, y, s)\) and a predefined search window of motion vectors \(V\), we calculate the mean squared error (or mean absolute error) between blocks \((x, y, s)\) in \(f_i\) and \((x', y', y', s)\) 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.

We then merge those adjacent blocks that have at least one motion vector in common in their initial set of motion vectors. This merging process helps solve the aperture problem because ambiguous motion within small blocks is refined as the blocks grow into a larger region. This also reduces the number of regions in the image and therefore the number of motion vectors. To simplify the merging process, we impose a complete quad-tree structure onto the small blocks and merge the small blocks into larger ones. In the full tree, there is a one-to-one correspondence between leaves and the initial square blocks. We define sibling blocks to be those blocks which share the same parent block.

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 \(i, j, k, l\) 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 is chosen to be the one associated with the minimum error of those within its intersection set.

The merging process is illustrated in Figure 2 which shows the initial sets of motion vectors for 16 blocks. For clarity, the diagram indicates only the direction of the candidate vectors and not also their magnitude. In the example shown, blocks 11, 12, 15 and 16 each have one motion vector which is common, and thus the intersection set of the larger block, having these four blocks as children, 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 vector chosen as that which provides the smallest total error of the two candidates. This is simply found by summing the residual error of the four blocks for each of the two matches. The remaining blocks in the example cannot be merged as the intersection sets of their parent blocks are empty. The motion vector for each child is
selected as the one associated with the minimum error of those within its initial set.

Fig. 2. Initial sets of motion vectors and merging of blocks.

The initial and intersection sets ($I_b$, $I_{sb}$) 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 machine-level bitwise logical AND operation.

The computational cost of this VSBM technique is little more than for fixed-size block matching. In fact, block matching is only performed at one level, and then the ‘minimal error’ tree is formed using a block merging process. This means that, computationally, the technique is not only very significantly less demanding than the optimal VSBM algorithm but also more efficient than other VSBM methods. Chan et al.\(^5\) for instance, estimated that from a large number of simulations on different sequences, their technique was two to three times as computationally demanding as fixed size block matching.

**4. PERFORMANCE EVALUATION**

We compare performance in terms of the mean square error (MSE) and present results for 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 bitrate codecs) and then processed only a centralised 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. An exhaustive search over $\pm 7$ pixels was adopted for all the tested techniques.

Figure 3 shows the result of applying each of the block matching motion estimation techniques to the ‘Miss America’ sequence. The MSE is given as the average value over 28 frames, which is considered more representative than presenting
Fig. 3. Comparative results over 28 frames of ‘Miss America’.

Fig. 4. Comparative results over 28 frames of ‘Split Screen’.

Respective results are presented in Figure 4 for the ‘Split Screen’ sequence. Here it is noticeable that the average MSE is much larger than for the ‘Miss America’ sequence. This is because the ‘Miss America’ sequence is relatively amenable to motion estimation as only a small number of areas move between frames. 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.

Fig. 5. Comparative results for 256 blocks per frame of ‘Split Screen’.

Fig. 6. Comparative results for 256 blocks per frame of ‘Foreman’.

It is appreciated that Figures 3 and 4 could hide aberrant behaviour if, for a prescribed number of blocks, there were large differences in MSE from frame to frame. However, this is not the case. Figure 5 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 frame sequence with no anomalous behaviour. Similar results are shown in Figure 6 for frames 40 to 70 of the ‘Foreman’ sequence. For both sequences the ‘bottom-up’ VSBM technique provides near-optimal results.
Figure 7 shows the same frame (from the ‘Split Screen’ sequence) which has been motion compensated using the optimal VSBM (OPT), bottom-up VSBM (ISECT) and top-down VSBM (TPDN) 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.

Figure 8 shows the optimal (OPT) and bottom-up (ISECT) VSBM 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.
5. CONCLUSIONS

In this paper, we investigated variable-sized block matching (VSBM) motion estimation techniques. They demonstrate considerable advantages over fixed-size block matching 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 localised or complex. VSBM also allows the total number of blocks in any frame to be varied while still representing true motion fairly accurately. This is important in codecs which adapt to maintain a constant bitrate. It also allows better bit allocation between the representation of motion and the residual (error) data.

We described an algorithm based on a quad-tree 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 codecs, 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 quad-tree based VSBM scheme.

We also described a new bottom-up VSBM technique which is as computationally efficient as fixed size blocking matching (FSBM) and yet provides near-optimal results. Block matching is performed only once, using small square blocks. Blocks are then merged in a quad-tree 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. Firstly, the computation is minimal as the search for matching blocks is no more demanding than for FSBM.

Secondly, the technique overcomes the ‘majority effect’ which is normally characteristic of top-down VSBM methods. In top-down ‘match or split’ algorithms, a large block may be considered matched when having an acceptable MSE, but a small area of the block may represent a feature with disparate motion. Essentially this is ignored because the MSE is biased by the majority of the pixels. This effect is not encountered in a bottom-up ‘match and merge’ algorithm. Bottom-up VSBM techniques should better represent the true motion within the image.

Our bottom-up algorithm also addresses the ‘aperture problem’. When a block represents a small component of a uniform moving feature, the inherent motion of the block can appear to be ambiguous. This results in a number of possible matches. Our algorithm addresses this problem by storing and subsequently intersecting vectors from adjacent blocks, and when the unique motion of a component of the feature is established the intersection set reduces to that vector representing the feature’s genuine movement.
Finally, 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 quad-tree based scheme. The residual error produced is significantly lower than for FSBM and also better than for top-down VSBM techniques.

6. ACKNOWLEDGMENTS

The authors are pleased to acknowledge the support for this work from the UK Science and Engineering Research Council, Grant No. GR/J79997.

7. REFERENCES

3. ISO/IEC IS 11172, "Information Technology - Coding of moving pictures and associated audio for digital storage media up to about 1.5 Mbit/s".