

Low Complexity Block Motion Estimation Using Morphological-based Feature Extraction and XOR Operations

*Thin M. Le, R. Mason, and S. Panchanathan

*School of Information Technology and Engineering, University of Ottawa, Ontario, Canada

email: thin@doe.carleton.ca

Department of Electronics, Carleton University, Canada

Department of Computer Science and Engineering, Arizona State University, USA

ABSTRACT

Motion estimation is a temporal image compression technique, where an $n \times n$ block of pixels in the current frame of a video sequence is represented by a motion vector with respect to the best matched block in a search area of the previous frame, and the DCT coefficients of the displaced block differences. In this paper, a low complexity technique for motion estimation is proposed. The proposed technique, first, reduces the n -bit grayscale frames into 1-bit binary frames using morphological filters, and determines the displacement of the edge features of the adjacent frames. While reduction in bit-depth requires a small percentage of computation using the full pixel resolution, the search procedure is performed by simple XOR logic operations and 1-b distortion accumulations on the entire search area. Compared to other low complexity techniques, the proposed technique yields better frame reconstruction, operates using simpler arithmetic/logic operations, and possesses a higher degree of parallelism when implemented on a 2-D Single-Instruction stream, Multiple-Data stream (SIMD) architecture.

Key words: motion estimation, morphological image processing, SIMD architecture, parallel processing.

1. INTRODUCTION

Motion estimation (ME) is a temporal image compression technique, where an $n \times n$ block of pixels in the current frame of a video sequence is represented by a motion vector with respect to the best matched block in a search area (SA) of the previous frame, and the DCT coefficients of the displaced block differences. Traditionally, there are two classes of motion estimation/compensation algorithms: pel-recursive [1] and Block Matching Algorithms (BMA's) [2]. Pel-recursive algorithms compute the displacement of each pixel individually. These algorithms do not require the transmission of motion information, but recursively use the luminance change to find the motion information. The advantage of pel-recursive algorithms is their ability to cope with multiple moving regions/objects and parts of a region/object undergoing different displacements. When pel-

recursive algorithms are employed for every pixel, the computation involved is extremely large. Therefore, they are often operated in a predictive manner, where the motion vectors between frame(t) and frame(t-1) are predicted using the corresponding motion vectors between frame(t-1) and frame(t-2).

BMA's, on the other hand, assume that all pixels within a block have the same motion. They perform efficiently with more relaxing conditions as described below [1]:

- Zooming and rotation of objects are not considered;
- Pixel illumination between frames is spatially and temporally uniform;
- Object displacement is constant within a small 2-D block of pixels; and
- Matching distortion increases monotonically as the displaced candidate block moves away from the direction of the exact minimum distortion.

Within BMA's, there are two sub-classes. The first sub-class includes Pixel Decimation (PD) [3], and its variant, 16:1 Subsampling BMA [4], both of which result in reduction in the number of pixels involved in each distortion computation by subsampling patterns. The second sub-class involves Three-Step Search (TSS) [5], and its variant, Simple and Efficient Search (SES) [6]; Conjugate Direction Search (CDS) [7]; and 2-D Logarithmic Search (2-D LS) [2], all of which focus on reducing the number of search locations in the SA using geometrical and logical deduction.

In the past, low complexity BMA's have also been presented in the literature [8], [9], [10], where the binary images corresponding to the original frames in the sequence are used for motion estimation. These techniques generally use either thresholding techniques or average filters to generate the binary frames from the grayscale pixel frames. A simple accumulate operation is then applied to identify the block with the most matching pixels or least distortions. The difference in these algorithms resides mainly in the techniques used to generate the binary frames.

In this paper, we propose a non-linear approach to low complexity BMA using morphological image processing for Feature Extraction and XOR operations (FEXOR) [11]. Morphological filters, when applied to continuous grayscale images, only require MAX (maximum) or MIN (minimum) operations on pre-defined windows. The proposed technique is robust and resource-efficient in generating a binary frame. The block motion vector is determined by comparing the displacement of the edge features of adjacent binary frames.

The original frame is first processed using the *opening-by-reconstruction* operation to filter out the noise, if any. Next, the *external-gradient* operation is applied to generate the edges. These morphological concepts will be reviewed in section 2. From an algorithmic view point, this technique is independent of the pixel illumination variations within a frame or between frames which is not the case with the BMA's. Besides, the speedup resulting from low complexity ME can be as high as 300 [10], [11], compared to full-search block matching algorithm (FBMA).

The remainder of the paper is organized as follows: A brief review of BMA's, followed by the presentation of FEXOR is provided in section 2. The study on computational complexity and resource utilization will be provided in section 3, followed by the performance analysis in section 4. The conclusions are provided in section 5.

2. REVIEW OF BLOCK MOTION ESTIMATION TECHNIQUES

2.1 Block Matching Algorithms

The full-search block matching algorithm (FBMA) [2] is commonly used as a benchmark for the performance comparison of other ME techniques. FBMA searches all possible displaced locations within an $n \times n$ SA to find the best match. Many matching criteria have been proposed to find the best match block [12]. These criteria include: the MSE (mean square error) and the MAE (mean absolute error). The MAE criterion is preferred because it requires no multiplication while yielding a similar performance compared to the MSE.

Let $X_{i,j}$ be the reference block at coordinates (i,j) , $Y_{i+k,j+l}$ be the candidate block at coordinates $(i+k, j+l)$, and p be the maximum displacement, the MAE distortion measure is given by:

$$MAE_{(k,l)}(X, Y) = \frac{1}{n \times n} \sum_{i=1}^n \sum_{j=1}^n |X_{i,j} - Y_{i+k,j+l}| \quad -p \leq k, l \leq p \quad (1)$$

For each of the $(2p+1)^2$ locations to be searched, the calculation of MAE requires $3n^2$ operations (that is one subtraction, one absolute operation, and one addition for each of the n^2 pixels in a block). Thus, for K reference blocks in frame(t), the computational complexity of FBMA is $O(3Kn^2(2p+1)^2)$. For an MPEG-2 [13] video frame, of size 720 x 576 pixels, with reference blocks of size 16 x 16 and a maximum displacement of 16 pixels, approximately 1.355 billion operations are required in the ME process.

In order to reduce the computational complexity of ME, within block matching algorithms, two main approaches have been

proposed. One reduces the number of pixels involved in each distortion computation, while the other focuses on reducing the number of search locations within the search area. The algorithms, which will be introduced shortly, and their variants (in brackets) are listed in Table 1 below:

Table 1: List of block matching algorithms

Approach	Name	Notes
I	PD (16:1 Subsampling)	reduce the number of pixels involved
II	TSS (SES), CDS, 2-DLS	reduce the number of search locations involved

The first approach includes Pixel Decimation (PD) [3] and 16:1 Subsampling BMA [4]. Liu *et al.* [3] have presented a full search algorithm, on the entire SA, where the pixels within each block, involved in the distortion computation process are 2:1 decimated in each direction, and the decimating patterns are alternated among the motion determination of adjacent reference blocks. Kim *et al.* [4] claim that faster estimation can be achieved, without significantly losing frame quality, by using a 16:1 decimating ratio.

The second approach includes Three-Step Search (TSS) [5], and its variant Simple and Efficient Search (SES) [6]; Conjugate Direction Search (CDS) [7], and 2-D Logarithmic Search (2-DL) [2]. In TSS, Koga *et al.* propose a divide-and-conquer search method where the SA is first divided into 9 sub-SA's. The sub-SA which results in the lowest MAE will be further divided and searched until a 3 x 3 search window is obtained, and the final motion vector is determined.

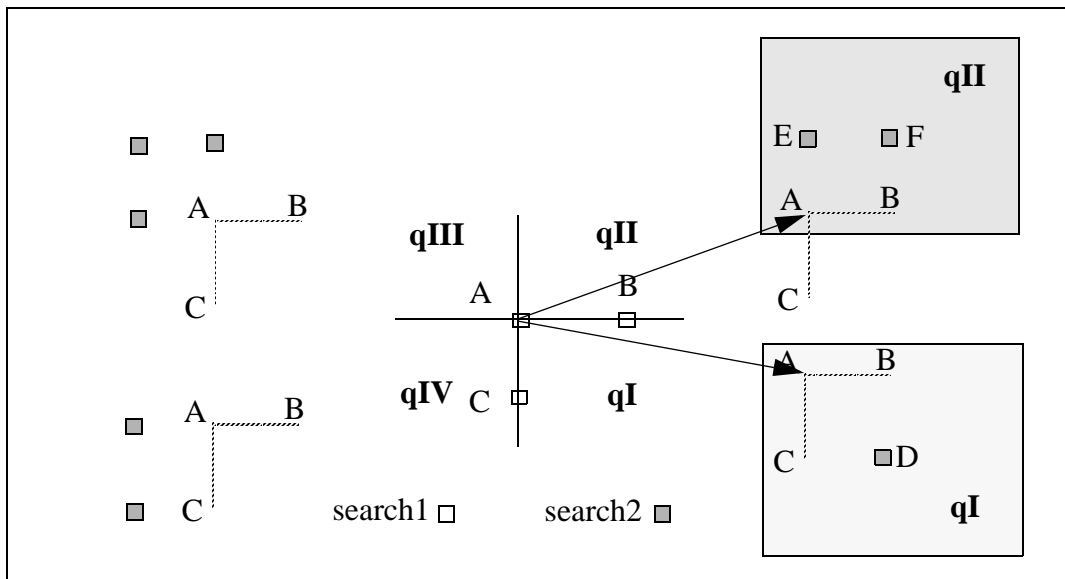


FIGURE 1: Simple and Efficient Search

Lu *et al.*'s technique in SES reduces the number of search locations by logically proceeding from the 3 starting points instead of the 9 sub-SA's. The original SA is divided into 4 quadrants (I, II, III, and IV), using 3 starting points A, B, and C, where A is the center of SA (Fig.1). The SES algorithm outlines that if $MAE(A) \geq MAE(B)$, then the best match is in the proximity of B in the horizontal direction. Also if $MAE(A) \geq MAE(C)$, then the best match is in the vicinity of C in the vertical direction. If the horizontal and vertical conditions are combined, in this case, quadrant I will be selected. The next search on quadrant I (whose exploded view is represented by the shaded area) will be based on the previously determined MAEs of points B and C, and the MAE of the new point D, where D is also the center of quadrant I. Therefore, one new MAE computation (for point D) is required. Had quadrant II been chosen, two MAE computations for E and F (center of quadrant II) will be required along with the previously determined MAE of point B. The SES, therefore, achieves an average speed-up factor of 2 compared to TSS.

In CDS, Srinivasan *et al.* propose a multiple 1-D search. Starting at point A, the search proceeds, location by location, in the horizontal direction. When the lowest MAE is obtained, at point B, the search proceeds in the vertical direction. When the lowest MAE is obtained, at point C, the diagonal connecting A and C will then be searched. The best match block is assumed to lie on the line connecting A and C.

In 2-D LS, Jain *et al.* present a coarse-to-fine moving search window in the North, East, West, and South directions. The search window is first moved in the direction of the smallest MAE. When the smallest MAE corresponding to the center of the search window is located, the search window is decreased by a factor of 2 in each dimension. The procedure is repeated until a 3 x 3 window is obtained where all 9 locations are searched. We note that, the former approach processes all pixel resolution of the pixels involved equally, while the later could potentially select the direction of the local minimum as the direction of the minimum distortion.

2.2 Low Complexity Block Matching Algorithms

Low complexity BMA's involve the computation of motion vectors from a binary map representing the blocks as opposed to the block grayscale. Gharavi *et al.* propose a technique where the pixels involved are classified into matched or mismatched pixels, and hence the name Pixel Difference Classification (PDC). A pixel is matched when the absolute difference between the pixel in the reference block and the corresponding pixel in the candidate block is less than a threshold. The candidate block with the largest number of matching pixels is chosen to be best match. In this technique, the selection of matched/mismatched

pixels primarily relies on a threshold which is subject to illumination changes within a frame or between frames. Feng *et al.* propose a Bit Plane Matching (BPM) technique, where the block mean is used as the threshold in contrast to an arbitrarily chosen threshold in PDC. If a pixel value is greater than the block mean, it is set to 1; otherwise, it is set to 0. Since block mean is required, there will be a blocking effect on the generated binary frames, hence, the estimation will be affected by the block boundaries. In Natarajan *et al.*'s technique, a 17 x 17 convolution kernel K which behaves like a band-pass filter, is used, where:

$$K_{i,j} = \begin{cases} \frac{1}{25} & \text{if } i,j \in [1, 4, 8, 12, 16] \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

If a pixel in the original frame is greater or equal to the corresponding pixel in the filtered frame, the resulting binary pixel is set to 1; otherwise, it is set to 0. This technique, referred to as Band-Pass Bit Plane Matching (BP_BPM), is superior to PDC and BPM. Therefore, BP-BPM will be selected for comparisons. However, from the implementation standpoint, it can be observed that:

- The larger the window, the larger the communication and control overheads;
- The further the distance between pixels, the lower the resource utilization.

In the following sections, a simple and vectorizable motion estimation FEXOR technique is proposed. We briefly present the basic concepts of mathematical morphology as applied to image processing.

2.3 Recalls on Mathematical Morphology

Mathematical morphology refers to a branch of non-linear image processing and analysis that exploits geometric structure of an image [14]. Mathematical morphology considers images as algebraic sets. Let A and B be two sets or images. The basic set operators on A and B are the union $A \cup B$ and the intersection $A \cap B$. These operators apply directly to binary images. For grayscale images, the corresponding operators are the supremum, $A \vee B$, and the infimum, $A \wedge B$. For digital data, the supremum and infimum are equivalent to maximum and minimum operators on a preset n x n pixel window, respectively.

To start with, the image under investigation is probed with a set of known shape called *structuring element* (SE). An origin must be specified for each SE. In this paper, the origin of the SE is at its center. We briefly introduce the two basic morphological operations: *erosion* and *dilation*.

The erosion of a set X by a SE(B) is denoted as $\epsilon_B(X)$ and is defined as the locus of points x such that B is included in X when its origin is placed at x :

$$\epsilon_B(X) = \{x | B_x \subset X\} \tag{3}$$

Dilation is the dual operator of erosion. The dilation of a set X by a SE(B) is denoted $\delta_B(X)$ and is defined as the locus of points x such that B hits X when its origin coincides with x :

$$\delta_B(X) = \{x | B_x \cap X \neq \emptyset\} \tag{4}$$

Erosion usually makes the set smaller than before, while dilation usually makes the set larger than before. Dilation of a previously eroded set does not allow, in general, recovery of the initial set. In fact, there exists no inverse transformations for erosion and dilation. The erosion and dilation operations are illustrated in Fig. 2.

Erosion and dilation are fundamental operations in morphological image processing. *Morphological opening* is defined as the erosion of a set using SE(B) followed by the dilation using SE(B^T), where SE(B^T) is the transpose of SE(B). If SE(B) is a square and symmetrical SE, then SE(B^T) is the same as SE(B). The erosion by SE(B) primarily removes objects (or noise) of sizes equal to or less than B. The subsequent dilation restores the shape of the remaining objects to some extent. Sometimes, the remaining objects become larger than their original sizes.

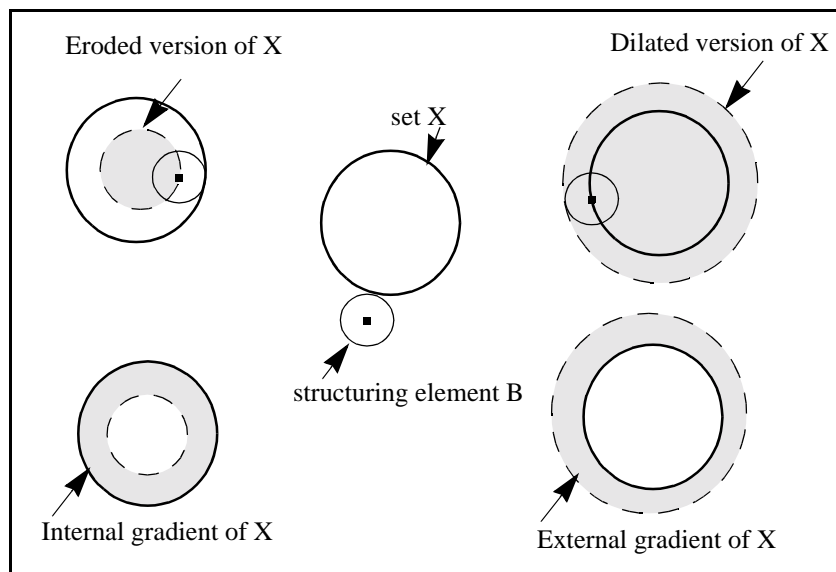


FIGURE 2: Erosion and Dilation with structuring element B

It is sometimes desirable to preserve the shapes of the remaining objects. Therefore, *opening-by-reconstruction* operation is required and is defined as the erosion of a set using $SE(B)$ followed by the dilation using $SE(B^T)$ with a condition. This condition ensures that the sizes of the remaining objects not to be larger than their original sizes after undergoing dilation. Therefore, if an object is not eliminated, its size and shape are preserved.

Once the shapes of the remaining objects are restored, their edges can be extracted using different techniques. If the eroded image is subtracted from the dilated image, then the resulting edge image is obtained by *morphological gradient* (MG). If the original image is subtracted from the dilated image, then the resulting edge image is obtained by *external gradient* (EG). Similarly, an image filtered by the *internal gradient* (IG) is obtained by subtracting the eroded version of an image from its original (bottom of Fig. 2).

2.4 Feature Extraction and XOR (FEXOR)

According to BMA's assumptions, the edges of the object/region in consecutive frames should be similar in shape if conditions 1, 2, and 3 are met. We explained how the edges of the object/region in a frame can be obtained by MG/EG/IG. This approach attempts to replace the intensive MAE computation on the full pixel resolution frames by the simple XOR operations and 1-b accumulations on the binary frames. There are two stages in FEXOR: 1-b transform and motion vector determination. In the 1-b transform stage, the grayscale images first undergo an opening-by-reconstruction operation¹ using a 3 x 3 window to filter out any noise. The resulting image then undergoes EG, using a 5 x 5 window. Finally, a threshold of 5 is applied² to generate a binary edge map.

In the motion vector determination stage, the FBMA technique is next applied on the entire SA of the edge maps. Since the edge maps are binary images, no full pixel resolution MAE computations are required. Instead, the XOR operations are performed at each search location. At each pixel in the search block, the result of an XOR operation is a 0 if the two pixels match, and 1, otherwise. The location resulting in the least number of 1's is the best match. Motion estimation using FEXOR

1. Closing-by-reconstruction can also be applied, but no significant improvement has been observed.

2. Dynamic thresholding can also be applied. The threshold is determined by halving the dynamically determined (MIN, MAX) range within a block or a row of pixels. The advantage of dynamic thresholding is that no prior knowledge of the image contrast is required.

generally results in a speed-up of nearly 8 (for 256 grayscale image) since the XOR operations and 1-b accumulations are much simpler than the 8-b MAE operations. PD or TSS may be applied instead of FBMA when further speed-up is desired.

From an implementation viewpoint, the 1-b transform stage involves point operations while the motion vector determination stage involves block operations. We note that the same set of operations are applied to each point (pixel) in the frame. Therefore, the SIMD architecture can be used. Similarly, the same set of instructions is also applied to each block in the frame. Thus, the same SIMD architecture can be employed again. In the following section, we will investigate the performance of FEXOR when implemented using an array of SIMD processors. Comparison with BP-BPM is also provided.

3. COMPUTATIONAL COMPLEXITY AND RESOURCE UTILIZATION

In this section, we will compare the proposed FEXOR technique against BP-BPM in terms of the computational complexity, evaluated via the number of single-bit operations. We also investigate communication issues to study the resource utilization if SIMD array processors are used. The low complexity ME techniques involve simple 1-bit operations to be applied to generate the binary frames. In order to have a fair comparison, 1-bit operations are chosen for the analysis.

3.1 Computational Complexity

FEXOR: There are 8 minimum/maximum operations per 3 x 3 erosion/dilation. For a 5 x 5 window, there are 24 min/max operations per erosion/dilation. The opening-by-reconstruction operation requires 1 erosion followed by 1 dilation. On the other hand, there are 1 dilation, 1 subtraction, and 1 thresholding operations per EG operation. The total of 1-b operations for 1-bit transform is: $8 \text{ bits} * (8\text{min.} + 8\text{max.} + 24\text{max.} + 1 \text{ sub.} + 1 \text{ thres.}) = 336$ 1-bit operations per pixel.

Motion vector determination: For each 16 x 16 pixel block, we need to search a total of $(2p+1)^2$ locations, where p is the maximum displacement. Since the search range is usually from -16 to +15, $(2p+1)^2$ is 1024. There are 256 XOR operations and 255 additions (whose operands range from 1 to 8 bits). An optimized addition algorithm of the 255 variable-length terms requires 502 1-b operations. Finally, there are 1024 8-bit comparisons to determine the closest match. The total number of 1-b operations per pixel in determining the motion vector is thus, $[1024*(1\text{-bit}*256 + 1\text{-b}*502) + 1024*8\text{-bit}]/16^2 = 3064$. Therefore, the total number of 1-b operations per pixel using FEXOR is 3400.

BP-BPM: There are 24 additions (whose operands range from 8 to 13 bits), one division (having 13-bit dividend and 5-bit divisor), and one 8-bit comparison. The total number of 1-bit operations per pixel at the 1-b transform stage is:

214 (for additions) + 120 (for division without overflow checking) + 8 (for comparison) = 342 1-bit operations.

The number of 1-b operations to determine the motion vector should be the same when using BP-BPM. Therefore, the total number of 1-b operations per pixel is 3404. It can be seen that the total number of 1-b operations per pixel in both techniques are comparable.

If multi-bit Arithmetic Logic Unit (ALU) is assumed, then from the circuit design viewpoint, the simpler operations of FEXOR like minimum/maximum search, addition, etc. in contrast to division for BP-BPM results in a smaller processing element, and thus more PE's can be realized on the same chip area.

3.2 Resource Utilization

Consider the implementation of FEXOR and BP-BPM techniques on a 2-D SIMD array processor with n^2 PE's. The question is which technique uses resources more efficiently compared to the other. Let us assume that the image is partitioned into blocks of $k \times k$ pixels ($k=16$) and that these blocks are going to be processed, one by one, on a $n \times n$ PE array processor ($n \geq 16$). If a 3×3 window is used, then in order to compute the results, the central PE will require one datum from the left PE, and one from the right PE. Therefore, at any one time, $(n-1)$ PE's are used in one dimension, and $(n-1)^2$ PE's are used in both dimensions. Similarly, a 5×5 window will result in 50% chance of $(n-1)^2$ and 50% chance of $(n-2)^2$ PE's being used in any one operation. For windows used in BP-BPM (see equation. 2), there are two cases: 50% chance of $(n-4)^2$ PE's being used if the pixels involved are 4 pixels apart, and 50% chance of $(n-8)^2$ PE's being used if the pixels involved are 8 pixels apart.

Using the ratio of the number of PE's being used at one time over the total number of PE's available in the system, we can determine the resource utilization factor corresponding to each technique. For FEXOR, recall that the frames are first eroded by a 3×3 window, and then dilated by the same size window. The resulting frames are next dilated by a 5×5 window. Finally, the frames are subtracted and thresholded. We note that subtraction and thresholding require all pixels, and hence all PE's participate in the computation. The numbers of PE's used are shown in Eq. 5 and Eq. 6, respectively:

$$P_{FEXOR} = 0.25(n-1)^2 + 0.25(n-1)^2 + 0.25(n-1)^2 + 0.25(n-2)^2 \quad (5)$$

$$P_{BP-BPM} = 0.5(n-4)^2 + 0.5(n-8)^2 \quad (6)$$

The resource utilization factors of FEXOR and BP-BPM are expressed in Eq. 7 and Eq. 8, respectively:

$$\eta_{FEXOR} = \frac{(n-1)^2 + (n-1)^2 + (n-1)^2 + (n-2)^2}{4n^2} = \frac{4n^2 - 10n + 7}{4n^2} = 1 - \frac{10n-7}{4n^2} \quad (7)$$

$$\eta_{BP-BPM} = \frac{(n-4)^2 + (n-8)^2}{2n^2} = \frac{2n^2 - 24n + 80}{2n^2} = 1 - \frac{12n - 40}{n^2} \quad (8)$$

For $n=16, 32, 64,$ and $128,$ we summarize the resource utilization factors in Table 2. We can see that if BP-BPM is implemented on a 16×16 PE array processor, it will be 44% under-utilized compared to that of FEXOR, assuming that a 16×16 window is chosen instead of the proposed 17×17 window. It can be seen from Table 2 that larger the number of PE's, the smaller the difference in resource utilization factor, since the distance between the PE's involved become small compared to the number of PE's. However, systems with large number of PE's are not popular and hence, very costly to implement, whereas systems with smaller number of PE's, for instance 16^2 or 32^2 can be practically realized. Therefore, in terms of resource utilization factor, FEXOR is superior to BP-BPM.

Table 2: Resource utilization factors in four 2-D n^2 SIMD array processor configurations

n	η_{FEXOR}	η_{BP-BPM}	under-utilized by
16	85.06%	40.63%	44.43%
32	92.36%	66.41%	25.95%
64	96.14%	82.23%	13.91%
128	98.06%	90.87%	7.19%

4. PERFORMANCE ANALYSIS

Simulations have been performed on the first 30 frames of 8 sequences: Garden and Table Tennis of sizes 720×480 pixels; Miss America and Salesman of sizes 352×288 ; Pingpong and Susie of sizes 352×240 ; Caltrain of size 512×400 ; and Trevor of size 256×256 . In these simulations, the frame being considered is divided into blocks of size 16×16 pixels, and the search range of $(-16,+15)$ pixels in each direction has been chosen. First, 3×3 windows are used for opening-by-reconstruction operation, then a 5×5 window for external gradient, and finally, a value of 5 for binary thresholding.

All simulation results are compared against the FBMA. In Table 3, the average PSNR of the reconstructed frames from frame 1 to 29 are listed. The second column shows the average PSNRs obtained using FBMA which serve as benchmarks for the comparisons. Miss America sequence appears to have the highest reconstructed quality since there is not a lot of motion in the sequence. On the other hand, the Pingpong sequence has the lowest quality since the sequence involves greater motion within its SA. The third column shows the average PSNRs obtained using BP-BPM. The best reconstructions, within a 1dB margin, relative to FBMA, are Garden, Table Tennis, Pingpong, and Trevor, while the worst reconstructions, outside the 2dB range, are Miss America and Salesman. The reasonably reconstructed sequences (from 1-2dB range) are Susie and Caltrain. The fourth

column shows the average PSNRs obtained using FEXOR. The closest reconstruction, within a 1dB margin, relative to FBMA, are Table Tennis, Salesman, and Trevor, while there is no reconstruction that falls outside the 2dB range. The reasonably reconstructed sequences (from 1-2dB range) are Garden, Miss America, Pingpong, Susie, and Caltrain.

Table 3: Low complexity performance comparison with respect to FBMA (in dB)

Average PSNR (dB)	FBMA	BP-BPM (degraded by)	FEXOR (degraded by)
Garden	27.13	26.33 (-0.80)	26.03 (-1.10)
Table Tennis	26.35	25.74 (-0.61)	25.36 (-0.99)
Miss America	37.79	35.44 (-2.35)	35.92 (-1.87)
Salesman	35.20	32.22 (-2.98)	34.22 (-0.98)
Pingpong	25.89	25.24 (-0.65)	24.79 (-1.10)
Susie	36.22	34.67 (-1.55)	34.83 (-1.39)
Caltrain	30.95	29.89 (-1.06)	29.55 (-1.40)
Trevor	34.98	34.32 (-0.66)	34.25 (-0.73)

The Table Tennis and Trevor sequences seem to have the required edges for low complexity motion estimation. On the Pingpong sequence, BP-BPM performs better than FEXOR. On the Salesman sequence, however, FEXOR is better than BP-BPM. The interesting fact to point out here is that BP-BPM degrades by nearly 3dB when tested on Salesman sequence. This can be attributed to the fact that the spacing of some patterns in the frame is equal to that of the filter defined by equation 2. Also, the BP-BPM technique does not respond well when tested using Miss America sequence.

In general, FEXOR can be applied to a wide range of sequences to estimate the motion vector and is better than BP-BPM in terms of reconstruction quality. The degradations of the reconstructed frames range, on an average, from 0.7 - 1.9dB using FEXOR, while as BP-BPM results in a larger degradation of 0.6 - 3.0dB.

Different entropies of the error frames obtained by FBMA, BP-BPM, and FEXOR are listed in Table 4 to demonstrate their respective performances in terms of bit rate. The entropies represent the amount of information which will be compressed in the subsequent DCT coders.

Table 4: Comparisons on the average entropies of the errors

Ave. entropy (bpp)	FBMA	BP-BPM (differed by)	FEXOR (differed by)
Garden	4.06	4.18 (+0.12)	4.16 (+0.10)
Table Tennis	4.54	4.60 (+0.06)	4.64 (+0.10)
Miss America	2.78	2.96 (+0.18)	3.03 (+0.25)
Salesman	3.12	3.23 (+0.11)	3.23 (+0.11)
Pingpong	3.55	3.62 (+0.07)	3.74 (+0.19)
Susie	2.87	3.06 (+0.19)	3.08 (+0.21)
Caltrain	3.42	3.53 (+0.11)	3.54 (+0.12)
Trevor	3.10	3.18 (+0.08)	3.18 (+0.08)

The comparable entropies of the error frames of BP-BPM and FEXOR imply that FEXOR does not compromise quality nor bit-rate for better resource utilization.

5. CONCLUSIONS

Low complexity motion estimation techniques for video compression, can result in a speed-up as high as 300 times [11], [10] compared to FBMA. In this paper, we have presented a morphological image processing technique, called FEXOR, for better performance and more efficient resource utilization compared to the existing low complexity techniques. We have shown that, FEXOR provides a better reconstruction quality (at most 2dB below FBMA) compared to BP-BPM whose worst case is nearly 3dB. We have also shown that the simpler operations of FEXOR like minimum/maximum search, addition, etc. in contrast to division operator in BP-BPM resulting in a smaller processing element design, and thus more PE's can be realized within the same chip area. Finally, when implementing FEXOR on array processors, the resource utilization is 7 to 44% better compared to BP-BPM for the same pixel processing time.

ACKNOWLEDGEMENT

The authors wish to thank Peter Nyasulu and the reviewers for their helpful comments and suggestions.

REFERENCES

- [1] A. N. Netravali and B. G. Haskell, *Digital Pictures: Representation and Compression*, Plenum Press, (1989).
- [2] J. R. Jain and A. K. Jain, "Displacement Measurement and Its Application in Interframe Image Coding", *IEEE Transactions on Communications*, vol. COM-29, no. 12, pp.1799-1808, (1981).
- [3] B. Liu and A. Zaccarin, "New Fast Algorithms for the estimation of block motion vectors", *IEEE Transactions on Circuits and Systems for Video Technology*, vol.3, no.2, pp.148-157, (1993).
- [4] Y. Kim, C.S. Rim, and B. Min, "A Block Matching Algorithm with 16:1 Subsampling and Its Hardware Design", *ISCAS'95*, pp.613-616, (1995).
- [5] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, and T. Ishiguro, "Motion Compensated Interframe Coding for Video Conferencing", *Proc. Nat. Telecom. Conf.*, pp-G5.3.1-5.3.5, (1981).
- [6] J. Lu and M. L. Liou, "A Simple and Efficient Search Algorithm for Block-Matching Motion Estimation", *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 7, no. 2, pp.429-433, (1997).
- [7] R. Srinivasan and K. R Rao, "Predictive Coding Based on Efficient Motion Estimation", *IEEE Transactions on Communications*, vol. COM-33, pp. 888-896, (1985).
- [8] H. Gharavi and M. Mills, "Blockmatching Motion Estimation Algorithms - New Results", *IEEE Transactions on Circuits and Systems*, vol. 37, no. 5, (1990).
- [9] J. Feng, KT. Lo, H. Mehrpour, and A.E. Karbowiak, "Adaptive Block Matching Motion Estimation Algorithm Using Bit-Plane Matching", *IEEE International Conference on Image Processing*, pp.496-499, (1995).
- [10] B. Natarajan, V. Bhaskaran, and K. Konstantinides, "Low-Complexity Block-based ME via One-Bit Transform", *IEEE Transactions on Circuits and Systems on Video Technology*, vol.7, no. 4, pp.702-706, (1997).
- [11] T.M. Le, M. Snelgrove, and S. Panchanathan, "Fast Motion Estimation Using Feature Extraction and XOR Operations", *Proc. SPIE Vol.3311 Multimedia Hardware Architecture '98*, pp.108-118, (1998).
- [12] C. H. Chou and Y. C. Chen, "A VLSI Architecture for Real-Time and Flexible Image Template Matching", *IEEE Transactions on Circuits and Systems*, vol. 36, no.10, pp.1336-1342, (1989).
- [13] ISO/IEC 13818-2, "Generic Coding of Moving Pictures and Associated Audio", Committee draft, Nov. 1993.
- [14] E.R. Dougherty, *An Introduction to Morphological Image Processing*, SPIE Optical Engineering Press, (1992).