Parallel Implementations of Block-Based Motion Vector Estimation for Video Compression on the MasPar MP-1 and PASM

Min Tan, Janet M. Siegel, and *Howard Jay Siegel*
Parallel Processing Laboratory
School of Electrical Engineering
Purdue University
West Lafayette, IN 47907-1285, USA

Abstract — *Video compression is important for video transmission and storage. This paper presents two parallel implementations of a compression technique called block-based motion vector estimation. Results from a 16,384 processor MasPar MP-1 (a SIMD machine) and the 16 processor PASM prototype (a partitionable SIMD/MIMD mixed-mode machine) are discussed. The trade-offs of using different modes of parallelism and different data distribution schemes are examined.*

1. Introduction

Compression techniques for video sequences have been the focus of research for video transmission and storage [4]. One well known technique is *block-based motion vector estimation* [2, 3], in which the current frame of a video sequence is divided into disjoint rectangular blocks of pixels. For each block in the current frame, a block of pixels in the previous frame is found that represents the closest match for it, according to some criterion such as the mean absolute difference (*MAD*). The spatial distance between the positions of the two blocks in their corresponding frames defines a two-dimensional motion vector associated with the block in the current frame. Because of the high redundancy that may exist between successive frames, an accurate estimation of the motion vectors makes compression possible. If maximum displacements of W_r pixels and W_c pixels are allowed for the row and column directions respectively, there are $(2W_r+1)(2W_c+1)$ locations to search in the previous frame for the closest match for each block in the current frame*.

Suppose the intensity of the pixel with coordinates (i,j) in video frame n is $I_n(i,j)$. Each video frame is divided into disjoint blocks with size $B_r \times B_c$, where B_r and B_c are the number of pixels in the row and column directions for each rectangular block, respectively. A block of $B_r \times B_c$ pixels is denoted by the coordinate (k, l) of its upper left corner pixel [5]. The size of the search subimage, which is referred to as the *displacement window* in this paper, is $(2W_r+B_r)(2W_c+B_c)$. The MAD between the block $(k, l)[n]$ of the current frame n and the block $(k+x, l+y)[n-1]$ of the previous frame $n-1$ is

$$\mathrm{MAD}[(k,l)[n], (k+x,l+y)[n-1]] = \left(\sum_i \sum_j |I_n(k+i, l+j) - I_{n-1}(k+i+x, l+j+y)|\right)/(B_r B_c),$$

where $0 \le i < B_r$ and $0 \le j < B_c$.

The motion vector $mv(k,l)$ of the block $(k,l)[n]$ is given by (mv_x, mv_y), such that

$$\mathrm{MAD}[(k,l)[n], (k+mv_x, l+mv_y)[n-1]] = \min\{\mathrm{MAD}[(k,l)[n], (k+x,l+y)[n-1]]\},$$

where $-W_r \le x \le W_r$ and $-W_c \le y \le W_c$.

Blocks at the edges of each frame require fewer operations for finding the corresponding motion vectors than do interior blocks, because the displacement window regions are constrained by image boundaries. The computational savings made possible by this property are easily realized using serial algorithms, but are more difficult to exploit using parallel algorithms. The choice of block and displacement window sizes predicate what type of parallel algorithm should be used.

A number of serial search algorithms have been proposed for block-based motion vector estimation (e.g., [3, 5]). To achieve real-time compression, a parallel implementation is desirable. The goal of this paper is to evaluate two parallel implementations, the rectangular and the row stripe subimage methods, using two parallel machines. The two machines are the MasPar MP-1 [6], an SIMD machine with 16,384 *PEs* (processing elements, i.e., processor/memory pairs), and *PASM* (*PA*rtitionable-*S*IMD/*M*IMD) reconfigurable parallel processing system small-scale prototype with 16 PEs in the computational engine [8].

2. Parallel Implementations

In the *rectangular subimage method*, it is assumed that N PEs are logically arranged as an $N_r \times N_c$ grid. The previous and current video frames are mapped onto the PEs by superimposing the two images onto the PE grid. Thus, if each frame is of size $M_r \times M_c$ (M_r and M_c are the number of pixels in the row and column directions, respectively), then each PE stores two $(M_r/N_r) \times (M_c/N_c)$ rectangular subimages. All PEs search for motion vectors simultaneously, each PE for one of its local blocks.

When searching the previous-frame subimage for the motion vectors of the blocks at the edge of the current-frame subimage, pixels from spatially adjacent previous-frame subimages need to be transferred. Assume that $M_r/N_r > W_r$ and $M_c/N_c > W_c$ (a reasonable as-

This research was supported by NRaD under subcontract number 20-950001-70. It used equipment supported by the National Science Foundation under grant number CDA-9015696.

*To be consistent with this terminology from the video compression field, X_r will be the number of items in a row (i.e., the number of columns) and X_c will be the number of items in a column (i.e., the number of rows).

sumption for this problem domain). PE i requires $M_c/N_c \times W_r$ pixels of the previous-frame subimage from the two PEs directly adjacent to it along its leftmost and rightmost columns and $M_r/N_r \times W_c$ pixels of the previous-frame subimage from the two PEs directly adjacent to it along its topmost and bottommost rows. PE i must also receive $W_r \times W_c$ pixels of the previous-frame subimage from each of the four PEs diagonally adjacent to it. Thus, a total of $2M_cW_r/N_c + 2M_rW_c/N_r + 4W_rW_c$ inter-PE data transfers are required per PE to find the motion vectors for all blocks of the current-frame subimage. All N PEs can transfer pixels of the previous-frame subimage simultaneously with appropriate network support (both the MasPar MP-1 and PASM have this type of network support).

If α is the time to perform an absolute difference calculation and an addition between two integers, γ is the time to perform a comparison between two integers, and τ is the inter-PE data transfer time for one pixel value, then the computation time using the rectangular subimage method is (detailed derivation is in [9])

$$M_rM_c(2W_r+1)(2W_c+1)(\alpha B_rB_c+\gamma)/(NB_rB_c). \quad (1)$$

The corresponding communication time is

$$(2M_cW_r/N_c + 2M_rW_c/N_r + 4W_rW_c)\tau. \quad (2)$$

It follows that the communication time depends on N_r and N_c. The best PE configuration depends on the video frame size (i.e., M_r and M_c) and the displacement window size (i.e., W_r and W_c) [9].

An alternate method for mapping the previous and current video frames among the N PEs is based on distributing consecutive rows of pixels and is referred to as the *row stripe subimage method*. Both previous and current $M_r \times M_c$ video frames are divided into N subimages of size $(M_r \times M_c/N)$. Each PE stores $M_r/B_r \times M_c/(NB_c)$ blocks of both the previous and current video frames.

The blocks close to the image border do not require a search of the whole displacement window space. In the row stripe method, no calculation is performed by any of the PEs on exterior column blocks for part of the displacement window search space, because these blocks are distributed evenly among all the PEs. By allowing all PEs to realize the savings on the exterior column blocks, the computation time is decreased. The rectangular subimage method cannot take advantage of this savings because the exterior blocks of the current video frame are distributed unevenly among the PEs. The PEs with no exterior blocks must search the whole displacement window space and will dominate the computation time required for the parallel implementation.

For the $2M_c/NB_c$ column border blocks of the current-frame subimage, the search space of the displacement window is $(W_r+1)(2W_c+1)$. Suppose W_r is a multiple of B_r and let $p = W_r/B_r$. For the $2M_c/NB_c$ blocks that are i $(0 \le i < p)$ blocks away from the column border blocks, the search space is $(W_r+iB_r+1)(2W_c+1)$. For the rest of the blocks, the search space is $(2W_r+1)(2W_c+1)$. Thus, the computation time using the row stripe subimage method is

$$(2W_c+1)(\alpha B_rB_c+\gamma)\, M_c/(NB_c) \times [(M_r/B_r)(2W_r+1) - W_r(p+1)]. \quad (3)$$

Compared with the computation time using the rectangular subimage method (shown in Eq.(1)), the time saved by taking advantage of the exterior blocks of the current video frame is

$$(2W_c+1)(\alpha B_rB_c+\gamma)M_cW_r(p+1)/(NB_c). \quad (4)$$

For the row stripe subimage method, PE i requires $M_r \times W_c$ pixels of the previous-frame subimage from PE $[(i-1) \bmod N]$ and $M_r \times W_c$ pixels of the previous-frame subimage from PE $[(i+1) \bmod N]$. It follows that the communication time is

$$2M_rW_c\tau. \quad (5)$$

This is in contrast to Eq. (2) for the rectangular subimage method.

Another method is based on distributing consecutive columns of pixels among the N PEs. This is referred to as the *column stripe subimage method*. Detailed discussion of the trade-offs between these two stripe methods is in [9].

Comparing the two different schemes for distributing the previous and current video frames, the rectangular subimage method usually requires fewer inter-PE data transfers compared with the row stripe subimage method. But the row stripe subimage method has the potential to decrease the computation time due to the reduced search space of the displacement window for the exterior column blocks. Because the computation time involved for each possible displacement position (i.e., $\alpha B_rB_c+\gamma$ for the calculation of the MAD) can be large when B_r and B_c are large, the computation time saved by using the row stripe subimage method can be significant. This trade-off between two data distribution schemes is analogous to that in [1].

3. Timing Results

The SIMD implementation on the MasPar MP-1 concentrates on the comparison between the estimated execution time (based on Eqs. (1, 2, 3, 5)). and the experimental values obtained by running the parallel programs on the machine. For both methods, the estimated values of both computation and communication times differ from the experimental data by at most 2.0% [9].

PASM is a partitionable SIMD/MIMD system concept being developed as a design for a large-scale dynamically reconfigurable parallel processing system [8]. A small-scale proof-of-concept prototype (16 PEs in the computational engine) has been built at Purdue University. PASM can be dynamically reconfigured to form submachines of various sizes. Each submachine can independently perform mixed-mode parallelism. PASM uses a flexible multistage interconnection network for inter-PE communication.

Fig. 1 shows the timings for SIMD, SPMD, and mixed-mode implementations of the rectangular subimage method ($M_r = 64$, $M_c = 128$, $W_r = 4$, $W_c = 6$, and $B_r = B_c = 4$) on PASM. *SPMD* (single program - multiple data stream) mode is a special form of MIMD mode

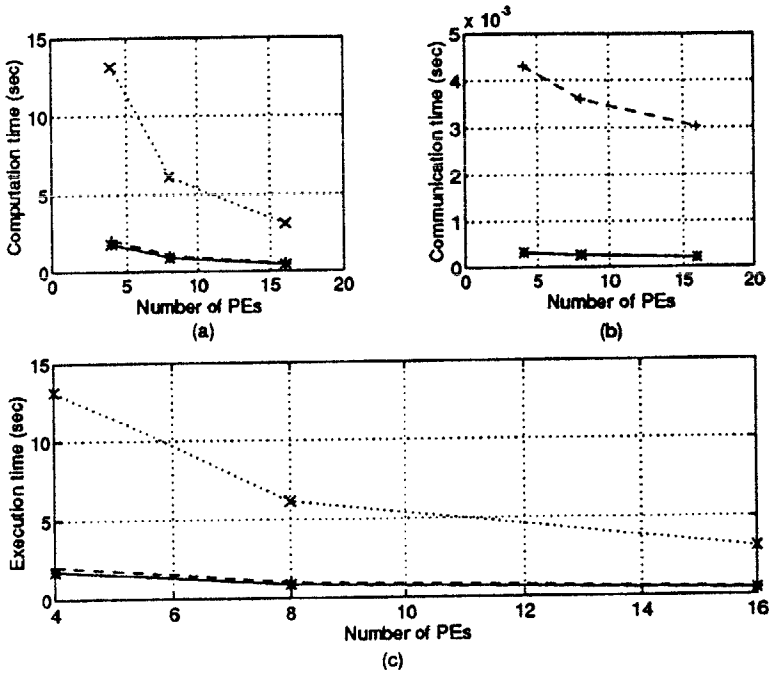

Fig. 1: The (a) computation time, (b) communication time, and (c) execution time for SIMD (dotted line), SPMD (dashed line), and mixed-mode (solid line) programs using the rectangular subimage method.

where all the PEs execute the same program in an asynchronous fashion, each on its own data. The three data points are for $N = 4$ ($N_r = N_c = 2$), 8 ($N_r = 4$ and $N_c = 2$), and 16 ($N_r = N_c = 4$).

As shown in Fig. 1(a), the SPMD mode implementation requires less computation time then the SIMD mode implementation. One of the advantages of SPMD mode is the ability to execute the "then" and "else" clauses of data conditional statements across different PEs simultaneously [7]. No PE is idle during the execution of a particular data conditional statement. In contrast, for SIMD mode, the "then" and "else" clauses of data conditional statements must be broadcast to PEs serially. Because finding the motion vector for each block of the current-frame subimage requires many comparisons (involved in α and γ), this part of the algorithm is implemented in SPMD mode in the mixed-mode implementation.

In the mixed-mode implementation, the loop control variables are stored and computed by the control unit (*CU*) in SIMD mode. Thus, the mixed-mode implementation can utilize CU/PE overlap, which occurs in this case when the CU performs the overhead associated with loop control in SIMD mode while the PEs execute the loop bodies in SPMD mode [7]. Because of this effect, the computation time for the mixed-mode implementation is somewhat less than the pure SPMD implementation.

One advantage of SIMD mode is the implicit synchronization that occurs after every instruction broadcast from the CU to the PEs, thus, communication protocols with less overhead than SPMD are used during inter-PE data transfers [7], as demonstrated by Fig. 1(b). Therefore, inter-PE communication is performed in SIMD mode in the mixed-mode implementation.

For both the rectangular and the row stripe subimage methods, the mixed-mode implementation has the smallest total execution time compared with the pure SIMD and pure SPMD implementations. Given this example, the total SPMD execution time is close to the mixed-mode time because the computation time (versus the communication time) dominates the execution. Other values for the machine and algorithm parameters could be chosen to show a larger difference.

Because the mixed-mode implementation performs best, all timing results for the rest of this section are for mixed-mode. The comparison between the estimated values for the timings of both data distribution schemes and the experimental values obtained by running the parallel programs on PASM is included in [9]. The estimated values of both computation and communication times (based on Eqs. (1, 2, 3, 5)) differ from the experimental data by at most 1.7%.

From the analyses shown in Section 2, the computation time for the row stripe subimage method is always smaller than the rectangular subimage method. This is illustrated by the experimental results shown in Fig. 2(a), in which $M_r = M_c = 64$, $W_r = W_c = 2$, and $B_r = B_c = 2$. In terms of the communication time, when N is increased, the rectangular subimage method has fewer inter-PE data transfers while the number of inter-PE data transfers stays the same for the row stripe subimage method (as shown in Fig. 2(b)).

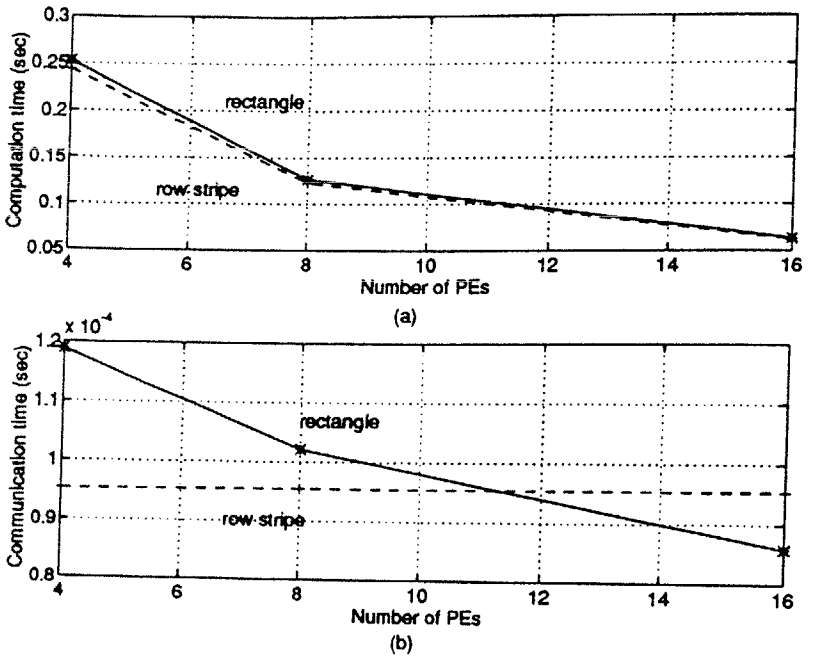

Fig. 2: The trade-offs between the rectangular subimage method (solid line) and the row stripe subimage method (dashed line); (a) computation time and (b) communication time.

4. Summary

This paper summarizes from [9] two parallel implementations of block-based motion vector estimation on two parallel machines. The timing results on the MasPar MP-1 and PASM demonstrate that this video compression technique can successfully exploit parallelism. The SIMD implementations on the MasPar MP-1 present an achievable speedup through the use of a commercially available massively parallel processing system.

Through the study of the mixed-mode implementation of this video compression technique on the PASM prototype, the trade-offs of using different modes of parallelism and different data distribution schemes are examined. The use of mixed-mode parallelism has some advantages over pure SIMD and pure SPMD implementations. With equivalent processing capability, a mixed-mode system can achieve higher speedup due to the flexibility of using more than one mode of parallelism.

References

[1] N. Giolmas, D. W. Watson, D. M. Chelberg, and H. J. Siegel, "A parallel approach to hybrid range image segmentation," *6th Int'l Parallel Processing Symp.*, Mar. 1992, pp. 334-342.

[2] "Coding of moving pictures and associated audio," *Committee Draft of Standard ISO11172*, Int'l Standards Organization/Motion Pictures Expert Group 90/176, Dec. 1990.

[3] J. R. Jain et al., "Displacement measurement and its application in interframe image coding," *IEEE Trans. Communications*, Vol. COM-29, Dec. 1981, pp. 1799-1808.

[4] D. LeGall, "MPEG: a video compression standard for multimedia applications," *Communications of the ACM*, Vol. 34, Apr. 1991, pp. 47-58.

[5] B. Liu et al., "New fast algorithms for the estimation of block motion vectors," *IEEE Trans. Circuits and Systems for Video Technology*, Vol. 3, Apr. 1993, pp. 148-157.

[6] J. R. Nickolls, "The design of the MasPar MP-1: a cost effective massively parallel computer," *IEEE Compcon*, Feb. 1990, pp. 25-28.

[7] H. J. Siegel, J. K. Antonio, R. C. Metzger, M. Tan, and Y. A. Li, "Heterogeneous computing," in *Handbook of Parallel and Distributed Computing*, A. Zomaya, ed., McGraw-Hill, 1995.

[8] H. J. Siegel, T. Schwederski, W. G. Nation, J. B. Armstrong, L. Wang, J. T. Kuehn, R. Gupta, M. D. Allemang, D. G. Meyer, and D. W. Watson, "The design and prototyping of the PASM reconfigurable parallel processing system," in *Parallel Computing: Paradigms and Applications*, A. Zomaya, ed., Chapman and Hall, London, U.K., 1995.

[9] M. Tan, J. M. Siegel, and H. J. Siegel, "Parallel implementations of block-based motion vector estimation for video compression on three machines," *Technical Report*, E.E. School, Purdue, in preparation.

Direct all inquiries to CRC Press, Inc., 2000 Corporate Blvd., N.W., Boca Raton, Florida 33431.

Catalog record is available from the Library of Congress
ISSN 0190-3918
ISBN 0-8493-2619-2 (set)
ISBN 0-8493-2615-X (vol. I)
ISBN 0-8493-2616-8 (vol. II)
ISBN 0-8493-2617-6 (vol. III)
ISBN 0-8493-2618-4 (ICPP Workshop)
IEEE Computer Society Order Number RS00027

Additional copies may be obtained from:

CRC Press, Inc.
2000 Corporate Blvd., N.W.
Boca Raton, Florida 33431