Sixth International Conference on Control, Automation, Robotic and Vision
5-8 December 2000, Marina Mandarin, Singapore

An Efficient Parallelization Scheme of Motion Estimation Algorithms on Myrinet-connected Workstations

Teddy Surya Gunawan, Stephanus Suryadarma Tandjung, Chong Man Nang

**School of Computer Engineering, Nanyang Technological University
Singapore 639798**

teddy@sentosa.sas.ntu.edu.sg, steph@ieee.org

***Abstract* — Motion algorithms have been developed for various applications such as image sequence analysis, machine vision, robotics, motion picture restoration or video coding. Motion estimation requires high processing requirements to estimate the image sequence with a reasonable frame size and quality. In this paper, we present an efficient parallelization scheme of motion estimation algorithms on Myrinet-connected workstations. Results indicated that using the proposed parallelization scheme, almost linear speedup curve could be achieved for four HDTV test video sequences experimented, indicating that if more processors are involved, less time will be required to process a frame.**

***Keywords* — Motion estimation, Myrinet-connected workstations, Parallel algorithms, HDTV image sequence**

1. INTRODUCTION

In recent years, there has been a growing interest in the area of image sequence processing especially video coding and video restoration. Motion estimation plays an important role in video coding and video restoration [1, 4, 10, 11]. Motion estimation obtains the motion compensated prediction by finding the motion field between the reference frame and the current frame

Motion estimation algorithm can be carried out using special-purpose hardware [5, 10]. However, a hardware-based approach has certain disadvantages. First, it is usually very expensive and ordinary users cannot afford it. Second, a hardware-based solution is less flexible and can become obsolete. Third, it is often optimized for a particular algorithm, and thus does not allow exploration of other present and future algorithms. Therefore, in order to overcome those disadvantages, a software-based solution is used.

A software solution using general-purpose computing platforms is more available. Furthermore, exploring new motion estimation algorithms is an active area of research, and software solutions have the flexibility to allow experimentation with such algorithms. However, the motion estimation algorithm has very high computation requirements especially when the image size is large, i.e. HDTV [3, 5, 14] size, or high quality is required. Therefore, instead of using single-processor or sequential computers, parallel computing is a promising solution for video coding and motion picture restoration.

A software-based implementation of motion estimation algorithm using multiple processors requires an efficient parallelization scheme. There have been some previous approaches [5, 8, 10, 12, 13, 17] on parallel video encoding and decoding, which includes motion estimation algorithms. A motion compensation for HDTV video encoder has been reported in [5] based on the block layer picture partitioning. Parallel MPEG-1 video encoding implemented on Ethernet-connected thirty workstations has been documented in [8]. An implementation of MPEG-2 video encoder on parallel and distributed systems is described in [13]. However, a parallel system has its communication and computation characteristics, in which the implementation of parallel algorithm on that particular system is still an active area of research.

This paper describes the parallelization scheme of motion estimation algorithms on Myrinet-connected workstations. The overview of motion estimation algorithms, include block-based motion estimation (ME), bi-directional ME (BME) and bi-directional hierarchical ME (BHME), are presented in section 2. Data distribution strategy and the parallel implementation of motion estimation algorithm are described in section 3. Section 4 describes the parallel platform. Section 5 includes experimental results, while the last section concludes the paper.

2. MOTION ESTIMATION ALGORITHMS

A number of very different motion estimation algorithms have been proposed in the literature. Detailed reviews are given by [1], [4], [6], [9], [18], [14], [19], [20]. These algorithms have been developed for various applications such as image sequence analysis, machine vision, robotics, image sequence restoration or image sequence coding.

Block-based motion estimation (ME) techniques are based on the minimization of a disparity measure. In order to alleviate prominent problems in the motion estimation algorithms, such as occlusion and scene changes, bi-directional motion estimation (BME) is employed. In this algorithm, three frames are used, including one preceding frame and one succeeding frame, to find a match between current frame and reference frame.

Hierarchical methods in motion estimation have been quite successful [1, 9, 14]. This algorithm is done recursively from higher layer to lower layer of the pyramids. The matching criterion used is:

$$MAD_l(x,y)=\frac{1}{\hat{M}\hat{N}}\sum_{i=0}^{\hat{M}-1}\sum_{j=0}^{\hat{N}-1}\left|I_{l,k}(i,j)-I_{l,k-1}(i+u,j+v)\right|$$
$$-D_l\le x,y\le D_l,\quad l=2,1,0 \qquad (1)$$

where $I_{l,k}(i,j)$ is the intensity of pixel at location (i,j) within the block in the kth frame at the lth level and the displacement is (u,v). $I_{0,k}(i,j)$ is the intensity of pixel on the full resolution image. $\hat{M}$ and $\hat{N}$ are height and width of a macroblock at level l, respectively. M and N are height and width of a macroblock at level 0. Notice that $\hat{M}=M/2^l$ and $\hat{N}=N/2^l$.

Using two times the motion vector found at level l-1 for level l as an initial vector, the motion vectors for level l are refined by using full search but with a relatively small search range. If the motion vector at level l-1 is represented by $V_{l-1}(x,y)$, the detected motion vector at level l can be described as

$$V_{l-1}(u,v)=2V_l(x,y)+\Delta V_{l-1}(\delta u,\delta v)\ ,\ l=2,1 \qquad (2)$$

where $\Delta V_{l-1}(\delta u,\delta v)$ is the updated increment of motion vector at level l-1 and is given below:

$$\Delta V_{l-1}(\delta u,\delta v)=\arg\min\frac{1}{\hat{M}\hat{N}}\sum_{i=0}^{\hat{M}-1}\sum_{j=0}^{\hat{N}-1}$$
$$\left|I_{l-1,k}(i+u,j+v)-I_{l-1,k-1}(i+u+\delta u,j+v+\delta v)\right|$$
$$-D_{l-1}\le\delta u,\delta v\le D_{l-1},\qquad l=2,1 \qquad (3)$$

Moreover, in order to further reduce the prediction error between the original image and the motion compensated image, the half-pixel search is implemented. The image with half pixel resolution is generated by interpolation from the original image, as follows:

$$S(u+0.5,v)=(S(u,v)+S(u+1,v))/2$$
$$S(u,v+0.5)=(S(u,v)+S(u,v+1))/2$$
$$S(u+0.5,v+0.5)=(S(u,v)+S(u+1,v)+S(u,v+1)$$
$$+S(u+1,v+1))/4 \qquad (4)$$

where u, v are the integer pixel horizontal and vertical coordinates, while S is the pixel intensity value.

The BME algorithm doubles the computational load by estimating backward and forward motion vector. To reduce the computation, we integrate BME and HME algorithms, called bi-directional hierarchical motion estimation (BHME). It has the advantage of handling such an occlusion problem while it reduces the complexity of motion vector estimation.

3. THE PARALLEL IMPLEMENTATION

In order to achieve a scalable parallel implementation of the motion estimation algorithms, we used the data-parallel or single program multiple data (SPMD) programming model. A single program was written for all processors that asynchronously execute this program on their local pieces of data. Communication of data and synchronization was done through message passing using MPICH [7] over GM [2] parallel programming environment.

The mapping of the ME algorithm is shown in Fig. 1, particularly for BHME algorithms (two other algorithms follow the mapping strategy described in Fig.1). The parallel programs start with initialization at every node, in which the computing environment for message-passing model is prepared. A frame of HDTV image is partitioned (using column stripe partitioning scheme [5]) and distributed to P processors. Laplacian pyramid is performed for each task to construct two lower resolutions of the original images. To produce motion vectors, the motion estimation is performed on each processor. Each processor sends motion vectors result to master processor. Finally, motion compensation in the master node is done to reconstruct the original image.

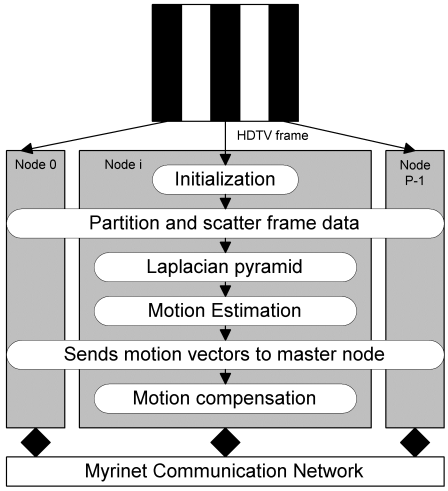

Fig. 1. Mapping of parallel motion estimation algorithms

For efficient parallel computing, the design of distributed data structures are of particular importance. This design should be made with a view to balance the computational load on individual processors and to gain maximum parallelism of interprocessor data communications. As a communication interface, we use Myrinet that can operate at 1.28 Gbit/s [2], and can transfer message 4.98 times faster than the Ethernet [21]. Basically, Myrinet is just like Ethernet. It is an add-on PCI card. Myrinet has its own driver in order to operate properly. This driver is bypassing the operating system call, so that it will not interrupt the operating system, and also will not interrupted by the operating system. In that scheme, it is possible to minimize the latency needed to send a message among processors, so that we can achieve higher data rate.

Fig. 1. shows that the interprocessor communications involved in parallel ME algorithms are the partition and sending of frame data and the collecting of motion vectors results to the master node. In our approach, we optimize those interprocessor communications to achieve maximum parallelism possible.

In our approach, a frame of size $P \times Q$ is partitioned into rectangular tiles of equal size $M \times N$, and each slice is assigned to one processor. The *MPI_Scatterv( )* function [7] was found to be the least communication time required to distribute the frame data, thus, this function is used in the implementation. Other interprocessor communication involved in motion estimation algorithm is the collection of the motion vectors result from slave nodes to master node. Instead of sending each motion vectors result directly to master node, we propose a scheme as shown in Fig. 2 to minimize the interprocessor communication time.

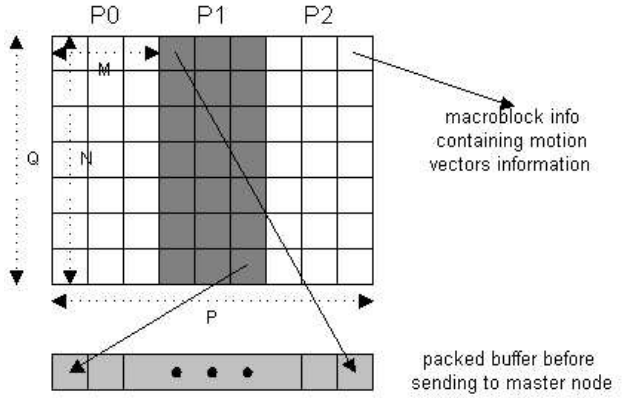

Fig. 2. The interprocessor communication scheme for sending the motion vectors results to master node

In this scheme, instead of transferring motion vectors results individually, these results are collated into a buffer and sent as a single message to the master node. Hence, the interprocessor communication time required is reduced from the total number of blocks in one frame into the number of processors involved in the parallel process.

4. System Configuration

The experimental apparatus is a cluster of 8 computers (with Intel Pentium II 450 MHz processor and 384 MB RAM) running Linux Redhat 6.0 with MOSIX 0.95 [16]. Each computer has a single Myrinet M2F-PCI32 network interface card on the PCI bus 32 bit-33 MHz, containing 1 MB SRAM card memory and a 37.5 MHz "LANai" processor [2]. The 8 computers are interconnected with one 8-port Myrinet switch M2F-SW8 through M2F-CB-10 Myrinet cables.

In this research, we use MPICH v1.1.2 on top Myrinet's GM v1.1.2 message passing library. MPICH is a public domain MPI [7] implementation developed at Argonne National Libraries and Mississippi State University. We configure MPICH over GM source using the "`-arch=LINUX –device=ch_gm`" option.

5. Experimental Results

Experiments were performed on the computing environment described in the previous section, using various numbers of processors. In this research work, we used four HDTV video sequences, including "basketball", "flag", "man", and "house". These sequences are representative of different kinds of motion and are very useful for testing motion estimation. The frames of the four-video sequences are displayed in Fig. 3. All the sequences are of the HDTV image size (1920×1080 pixels).

The measured computation time (parallel execution time) was averaged over n frames of a video sequence (n=30 for basketball sequence, and n=40 for the others). The time to process n frames of a video sequence was not necessarily the same in each processor, therefore the average was also taken over all the processors. Moreover, all of the timings were measured with microsecond precision using *MPI_Wtime( )* function.

We used the macroblock size of 16×16 pixels and the search area of 32×32 pixels. The exhaustive search is employed to achieve the best picture quality. In order to measure the quality of the video, we used the peak signal-to-noise ratio (PSNR) and entropy. The average values of PSNR obtained for different sequences over n frames of the three parallel motion algorithms are shown in Table 1. Of the three algorithms, the BME algorithm was found to have the highest quality. However, the kind of motion of the sequence does affect the results, in which the sequence that has low motion activity tends to achieve high PSNR.

Note that there is no performance loss due to parallelization since, in our programming model, the motion estimation algorithm in each processor uses the same computational logic that is used in the sequential version. Therefore, the magnitude of PSNR is dependent on the motion estimation algorithm itself.

Fig. 4, Fig. 5, and Fig. 6 show the speedups of the three motion estimation algorithms for the various HDTV video sequence. The linearly increasing curves for the modules show that use of more processors will result in greater speed of processing a frame. However, results show that different sequence has different speedup. Clearly, the findings indicate that the kind of motion has an effect to the speed of processing, in which the sequence that has low motion activity, i.e. "house" sequence, requires smaller processing time, vice versa. Thus, the speedup is affected by the kind of motion.

We notice also that all three parallel motion estimation algorithms implemented in this work have a comparable speedup. The interprocessor communication scheme proposed in section 3 tends to minimize the ratio of the communication time to the total computation time. Therefore, we can achieve almost linear speedup

curve for all three algorithms implemented, indicating that if more processors are involved, less time will be required to process a frame.

Finally, table 2 shows the frame rate using various numbers of processors. It shows that as the number of processors increases, the parallel motion estimation system is able to process more frames per second. The parallel BHME achieved higher frame rate with reasonable picture quality compared to other algorithms.

6. CONCLUSIONS

We have presented the implementation and performance of parallel motion estimation algorithms on Myrinet-connected workstation. Harnessing the communication power of Myrinet, three algorithms were implemented including ME, BME, and BHME. However, the interprocessor communication scheme involved in motion estimation algorithms, such as sending a frame data and collecting motion vectors results, need to be carefully designed to minimize the communication overhead.

Four HDTV test video sequences were used for assessing the performance of parallel implementation of the three motion estimation algorithms. The performance was evaluated in terms of speedup and frame rate. Moreover, the picture quality in terms of PSNR and Entropy was presented. Almost linear speedup curve was achieved for all three parallel implemented algorithms. One possible conclusion is that the interprocessor communication scheme proposed has successfully contributed to the results achieved.

REFERENCES

[1] M. Bierling, "Displacement estimation by hierarchical blockmatching," *SPIE Visual Comm. Image Process.*, Vol. 100, pp. 942-951, 1988.

[2] N.J. Boden, D. Cohen, R.E.Felderman, A.K.Kulawik, C.L.Seitz, J.N.Seizovic, and W-K.Su, "Myrinet : a gigabit-per-second Local Area Network," *IEEE Micro*, Vol. 15, pp. 29-36, February 1995.

[3] F. Bellifemine, A. Chimienti, and R. Picco, "Evolution and trends of HDTV," in *CompEuro '91 Advanced Comp. Tech.*, pp. 155-163, 1991.

[4] S. Kalra, M. N. Chong, "Bidirectional motion estimation via vector propagation," *IEEE Trans. Circuits and Systems for Video Tech.*, Vol. 88, pp. 976-987, Dec. 1998.

[5] Charng L. Lee, "Parallel implementation of motion-compensation for HDTV video decoder," in *Proc. of IEEE Int. Symp. on Consumer Electr.*, pp. 51-54, 1997.

[6] F. Dufaux, F. Moscheni, "Motion estimation techniques for digital TV : a review and a new contribution," in *IEEE Proc.*, Vol. 83, pp. 858-876, June 1995.

[7] W. Gropp, E. Lusk, and A. Skjellum, *Using MPI*, MIT Press, Cambridge, MA, 1994.

[8] Jongho Nang and Junwha Kim, "An effective parallelizing scheme of MPEG-1 video encoding on ehernet-connected workstations," in *Proc. Advances in Parallel and Distributed Computing*, pp. 4-11, 1997.

[9] J. Konrad, "Motion detection and estimation," in Image and Video Processing Handbook (A. Bovik, ed.), chap. 3.8, Academic Press, 1999.

[10] M. N. Chong, S. Kalra, and D. Krishnan, "Video restoration on a multiple TMS320C40 system," Texas Instrument (TI) Application Report, TI Incorp., Houston, TX, Nov. 1996.

[11] Anil K. Kokaram, "Motion Picture Restoration", Ph. D. Thesis, Dept. of Eng., Univ. of Cambridge, May 1993.

[12] Ioannis Pitas, *Parallel Algorithms for Digital Image Processing, Computer Vision and Neural Networks*, John Wiley & Sons, 1993.

[13] Shahriar M. Akramullah, Ishfaq Ahmad, and Min L. Liou, "Performance of software-based MPEG-2 video encoder on parallel and distributed systems," in *IEEE Trans. on Circuits and Systems for Video Technology, Vol. 7*, no.4, pp 687-695, August 1997.

[14] K. M. Uz, M. Vetterli, D. LeGall, "Multiresoultion approach to motion estimation and interpolation with application to coding of digital HDTV," in *Proc. ISCAS-90*, pp. 1298-1301, May 1990.

[15] V. Bhaskaran and K. Konstantinides, *Image and video compression standards*, Kluwer Academic Pub, 1995.

[16] A. Barak, S. Guday, and R.G. Wheeler, *The MOSIX Distributed Operating System: Load Balancing for UNIX*, in *Lecture Notes in Computer Science*, Vol. 672, Springer-Verlag, 1993.

[17] A.N. Choudhary, M.K.Leung, T.S.Huang, and J.H. Patel, "Parallel implementation and evaluation of motion estimation system algorithms on a distributed memory multiprocessor using knowledge based mappings," in *10th Int. Conf. on Pattern Recog.*, Vol. 2, pp. 337-342, 1990.

[18] C. Stiller and J. Konrad, " Estimating motion in image sequences: a tutorial on modeling and computation of 2D motion," in *IEEE Signal Processing Magazine*, pp. 70-98, July 1999.

[19] S. S. Tandjung, T. S. Gunawan, and M. N. Chong, "Motion estimation using adaptive matching and multiscale methods", to be presented in *Int. Conf. on Visual Comm. and Image Proc.*, June 2000.

[20] S. S. Tandjung, T. S. Gunawan, and M.N.Chong, "Motion estimation using adaptive blocksize observation model and efficient multiscale regularization", to be presented in *Int. Conf. on Image Processing*, September 2000.

[21] T. S. Gunawan, S. S. Tandjung, and M. N. Chong, "Performance of the Communication Layer with the Myrinet Gigabit LAN for HDTV Images on Pentium-Linux cluster using MPI", in *Proceeding IECI MULNET* 2000, pp. 255-259, April 2000.

Table 1. The PSNR and the Entropy of the test video sequence

Video Sequence	Parallel ME		Parallel BME		Parallel BHME	
	PSNR (dB)	Entropy (bits)	PSNR (dB)	Entropy (bits)	PSNR (dB)	Entropy (bits)
Basketball	28.47	3.68	29.22	4.08	28.98	3.72
Flag	33.34	11.04	34.94	11.06	27.93	6.76
Man	37.44	5.99	41.47	6.26	34.90	4.34
House	40.56	1.31	42.80	0.77	42.44	0.63

(a) Basketball

(b) Flag

(c) Man

(d) House

Fig.3. The HDTV video test sequences

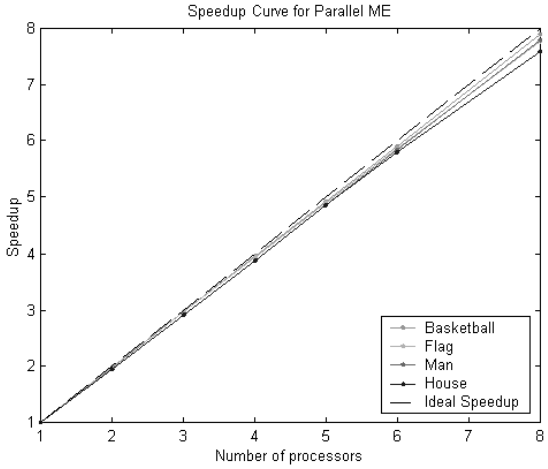

Fig. 4. Speedup of Parallel ME

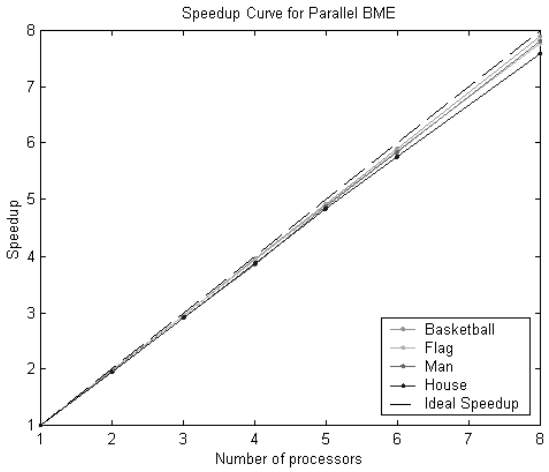

Fig. 5. Speedup of Parallel BME

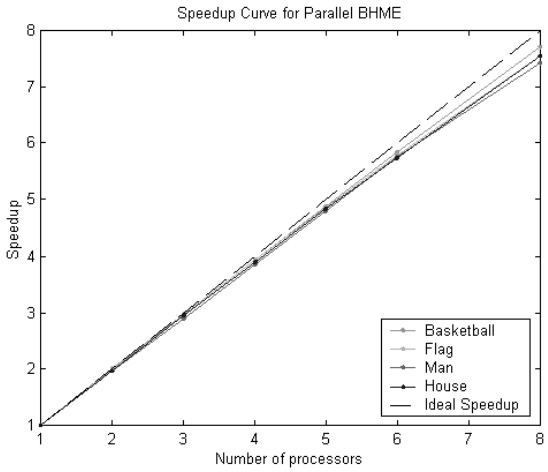

Fig. 6. Speedup of Parallel BHME

Table 2. Frame rate over various numbers of processors (frame/seconds)

Video Sequence	Parallel ME				Parallel BME				Parallel BHME			
	1	2	4	8	1	2	4	8	1	2	4	8
Basketball	0.004	0.009	0.017	0.035	0.002	0.004	0.009	0.017	0.020	0.040	0.079	0.154
Flag	0.005	0.010	0.019	0.037	0.002	0.005	0.009	0.019	0.023	0.045	0.090	0.172
Man	0.005	0.009	0.017	0.035	0.002	0.004	0.009	0.018	0.021	0.041	0.081	0.156
House	0.004	0.009	0.017	0.034	0.002	0.004	0.009	0.017	0.020	0.040	0.080	0.155