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

Parallel Implementation of Adaptive Bi-directional Motion Estimation based on Texture Analysis

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* — Various motion algorithms have been developed for various applications such as image sequence analysis, machine vision, robotics, motion picture restoration or video coding. In this paper, the parallel implementation of bi-directional motion estimation (ME) algorithm on Myrinet-connected workstations is presented. The bi-directional ME is based on texture analysis of the frame difference signals. Texture analysis is adopted here to reduce the computational loads of ME without compromising on the motion vectors quality. The bi-directional scheme and half-pixel search are employed to achieve higher picture quality. The performance of the proposed ME is compared with the classical motion algorithms on four HDTV video sequences.**

***Keywords* — Adaptive motion estimation, Myrinet-connected workstations, Parallel algorithms, Texture analysis, HDTV**

1. INTRODUCTION

Various motion algorithms have been developed for various applications such as image sequence analysis, machine vision, robotics, motion picture restoration or video coding [13]. Motion estimation (ME) obtains the motion field between the reference frame and the current frame.

The ME algorithm is computationally expensive and the cost increases exponentially with the image size [3, 5, 17]. Various fast algorithms [13] have been proposed to reduce the computational load. However, the conventional fast methods often converge to a local minimum due to their intrinsic selective nature. The information extracted from the texture analysis of the frame difference signal can be used to alleviate these deficiencies [1, 2, 8, 9, 10].

ME algorithms can be carried out using special-purpose hardware [17]. However, a software solution using general-purpose computing platforms is more available. Exploring new ME algorithms is an active area of research, and software solutions have the flexibility to allow experimentation with such algorithms. In addition, instead of using single-processor or sequential computers, parallel computers can be used to distribute the computational load to achieve higher throughput.

A software-based implementation of ME algorithm on multiple processors requires an efficient parallelization scheme. Parallel video encoding and decoding algorithms, which include ME algorithms, have been reported in [16, 17, 18, 19]. A motion compensation scheme 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, each parallel system has its unique communication and computation characteristics and investigation need to be carried out for the design of more efficient parallel ME algorithms.

This paper describes the parallelization scheme on Myrinet-connected workstations. The overview of ME algorithm based on texture analysis is presented in section 2. The parallel implementation of ME algorithm and the computing surface are described in Sections 3 and 4. Section 5 presents our experimental results, while the last section concludes the work to date.

2. INTERFRAME TEXTURE ANALYSIS AND ADAPTIVE ME ALGORITHMS

Texture is one of the important characteristics used in identifying objects or regions of interest in still images, and it can be defined by a set of statistics extracted from the local picture property [20]. The texture analysis of the frame difference signals provided to ME algorithms have been proposed in the literature [1, 2, 8, 9, 10]. However, of the various approaches, the method proposed in [1] is chosen because of it reduces the computational load without compromising on motion vectors quality. In an attempt to achieve higher picture quality, the bi-directional approach [14] is applied in our implementation of the ME algorithm presented in [1].

The temporal difference histogram is derived from the absolute difference of the gray level values between pairs of pixels belonging to successive frames ($g_1(x, y, t_1)$ and $g_2(x, y, t_2)$) of video sequence.

A pixel displacement from the previous frame by Δx and Δy, is represented by $\delta=(\Delta x, \Delta y)$ and called the *intersample spacing vector*.

For a particular value of δ, the absolute temporal difference function $f_\delta(x, y)$ is given by:

$$f_\delta(x,y) = \left| g_2(x,y,t_2) - g_1(x+\Delta x, y+\Delta y, t_2) \right| \quad (1)$$

The *temporal difference histogram*, p_δ is then defined as the probability density of $f_\delta(x, y)$, as shown below:

$$p_\delta(i) = \frac{1}{N_x N_y} \sum_{y=0}^{Ny-1} \sum_{x=0}^{Nx-1} h(x,y)$$
$$h(x,y) = \begin{cases} 1 & if\ f_\delta(x,y) = i \\ 0 & if\ f_\delta(x,y) \neq i \end{cases} \quad (2)$$

where N_x and N_y are the number of pixels in the x and y direction respectively.

Of the various features that can be derived from the temporal difference histogram, the contrast and inverse difference moment were chosen in [1]. For a block size of $N \times N$ pixels, with *M-level* quantization resolution, the *temporal contrast*, *TCON*, is defined as the moment inertia of p_δ around the origin, given by:

$$TCON = \sum_{i=0}^{M-1} i^2 p_\delta(i) \quad (3)$$

TCON gives a quantitative measure for the coarseness of the texture and its value depends on the amount of local variations present in the block. *TCON* must be normalized by a measure corresponding to the size and texture of the moving object. Hence, *local contrast*, *LCON* is employed.

$$LCON = \frac{1}{MA} \sum_{MA} [g(x,y) - \bar{g}]^2 \quad (4)$$

where $\bar{g}$ is the average gray value of the *moving area*, *MA*, within the pixel block. Based on the temporal and local contrast (Eq. (3) and Eq. (4)), a good estimate of the *average motion speed*, *S*, within a block can be defined as:

$$S = k \frac{TCON}{LCON} \quad (5)$$

where *k* is a constant with empirically selected values. Large values of *S* are expected in fast moving objects with uniform texture, vice versa.

The *inverse difference moment*, *IDM*, is a measure of homogeneity of an image and is given by:

$$IDM = \sum_{i=0}^{M-1} \frac{p_\delta(i)}{1+i^2} \quad (6)$$

IDM is used to identify the principal texture direction. Texture directionality can be analyzed by comparing spread measures of p_δ for various directions of δ, as shown in Fig. 1. The *IDM* can be used to estimate the directionality of the texture. The maximum value of *IDM* indicates that the frame difference signal is more homogeneous in that direction than others.

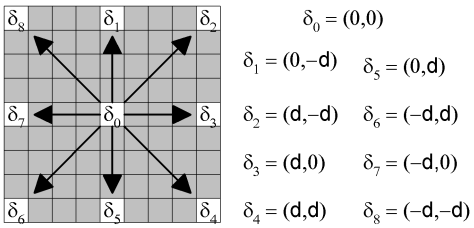

Fig. 1. The intersample spacing vectors

For each block, nine temporal difference histograms, $p_{\delta i}$ with i = 0, 1, …,8 were obtained. The δ_i grid is shown in Fig. 1, where *d* is the intersample spacing distance (ISD). The *TCON* and *LCON* are calculated from the $p_{\delta 0}$ histogram, using (3) and (4). The *IDM* values are calculated from the rest of $p_{\delta i}$ histograms. The motion speed, *S*, is calculated using (5) with *k=3*. Then the minimum and maximum displacements, *MinDisp* and *MaxDisp*, can be calculated empirically as:

$$MaxDisp = \begin{cases} 2(S+2) & \text{if } S = 0,1,2 \\ 8 & \text{if } S > 2 \end{cases} \quad (7)$$
$$MinDisp = MaxDisp/4$$

The maximum value of the IDM is computed to find the texture directionality. That is:

$$IDM_{\max} = \max\{IDM_{\delta i}\}, \qquad i = 1,2,...,8 \quad (8)$$

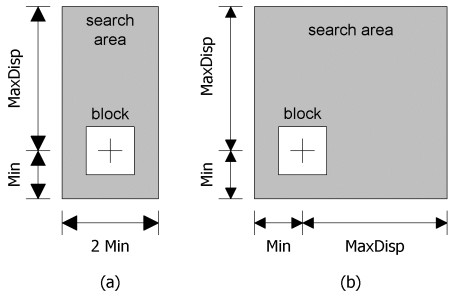

Fig. 2. Search area defined by the adaptive algorithm: (a) For vertical motion upwards, (b) For diagonally motion to northeast

If $IDM_{\max}$ corresponds to a vertical or a horizontal direction, let the search length towards this direction be equal to *MaxDisp* and the search widths towards any other direction be equal to *MinDisp*, as shown in Fig. 2. Here the search area changes adaptively, in size and shape, depending on the value of the speed, *S*, and the direction resulted from the inverse difference moment, *IDM*.

To further reduce the prediction error between the original image and the motion compensated image, the half-pixel search is implemented. Moreover, to alleviate prominent problems in the ME algorithms, such as occlusion and scene changes, bi-directional motion

estimation (BME) is employed, as shown in Fig. 3. 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. The texture analysis is done on two temporal difference frames, as shown in Fig. 3.

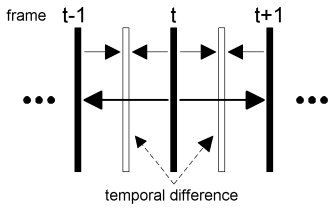

Fig. 3. Bi-directional ME based on texture analysis

3. THE PARALLEL IMPLEMENTATION

In order to achieve a scalable parallel implementation of the ME 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 [6] over GM [4] parallel programming environment. The mapping of the ME algorithm based on texture analysis is shown in Fig. 4.

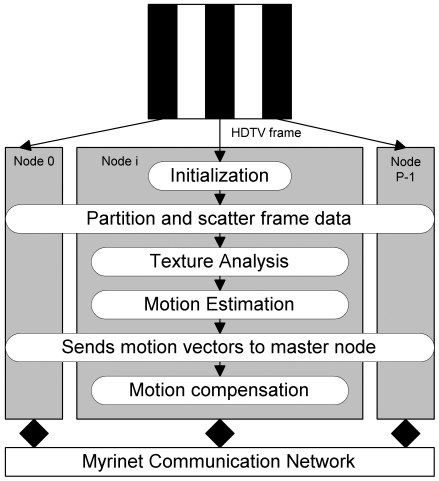

Fig. 4. Mapping of parallel texture ME algorithms

Fig. 4 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, 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 [6] was found to be the least communication time required to distribute the frame data, thus, this function is used in the implementation. Moreover, 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 minimized.

4. SYSTEM CONFIGURATION

The equipment used in our experiments is a cluster of 8 Linux workstations (with Intel Pentium II 450 MHz processor and 384 MB RAM) running Linux Redhat 6.0 with MOSIX 0.95 [7]. Each node has a single Myrinet M2F-PCI32 network interface card on the PCI bus 32 bit, containing 1 MB SRAM card memory and a 37.5 MHz "LANai" processor [4]. The machines 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 [6] 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 carried out using the computing resources described. In this research work, we used four HDTV (1920×1080 pixels) video sequences, as shown in Fig. 5, including "basketball", "flag", "man", and "house" sequence. These sequences are representative of different kind of motion and are very useful for testing the capability of ME algorithms. To compare the performance, three ME algorithms were parallel implemented, namely, block-based ME, ME based on texture analysis [1] (TME), and bi-directional ME based on texture analysis (BTME).

The measured computation time (parallel execution time) was averaged over n frames of each 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 video sequences. All the timings were measured with microsecond precision using *MPI_Wtime( )* function [6].

We used the macroblock size of 16×16 pixels and the search area adaptively defined by texture analysis of the frame difference signals. The average values of PSNR and Entropy obtained for different sequences over n frames of the three parallel motion algorithms are shown in Fig. 7 and Fig. 8, respectively.

As described in [1], an important parameter that affects the performance is the intersample spacing distance value (d value). To find the best d values for particular sequence, various d values from 1 to 10 were experimented. The type of motion for particular sequence affects the best d value achieved, as shown in Fig. 6.

Note that there is no performance loss due to parallelization since, in our programming model, the ME 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 ME algorithm itself.

Table 1 and Table 2 show the speedup and the frame rate of the three ME algorithms implemented for the various HDTV video sequence. The linearly increasing speedup shows that use of more processors will result in greater speed of processing a frame. Although that there is not much improvement in the motion vector results for the TME and BTME algorithm, but we can see that the frame rate is higher. The frame rates of TME and BTME algorithm are faster 17 and 8 times respectively than the frame rate of ME algorithm. Nevertheless, there will always a tradeoff between the high quality picture and the speed of computation.

6. CONCLUSIONS

We have presented the implementation and performance comparison of adaptive bi-directional ME algorithm based on texture analysis on Myrinet-connected workstation. Four HDTV test video sequences were used for assessing the performance of parallel implementation of the three ME algorithms, namely ME, TME, and BTME. The performance was evaluated in terms of speedup and frame rate. Moreover, the picture quality in terms of PSNR and Entropy was presented. As expected, the parallel BTME achieved higher frame rate with reasonable picture quality compared to other algorithms.

REFERENCES

[1] V. E. Seferidis and M. Ghanbari, "Adaptive Motion Estimation Based on Texture Analysis", *IEEE Trans. on Comm.*, Vol. 42, pp. 1277-1287, April 1994.

[2] V. Seferidis and M. Ghanbari, "Use of Co-occurrence Matrices in the Temporal Domain", *Electronics Letters*, Vol. 26, No. 15, pp. 1116-1118, July 1990.

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

[4] 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*, 15(1):29-36, February 1995.

[5] T.S. Gunawan, S.S.Tandjung, 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

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

[7] 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.

[8] K. Otsuka, T. Horikoshi, S. Suzuki, and M. Fujii, "Feature extraction of temporal texture based on spatiotemporal motion trajectory", in *Int. Conf. on Pattern Recog.*, Vol. 2, pp.1047-1051, August 1998.

[9] K. Otsuka, T. Horikoshi, and S. Suzuki, "Image velocity estimation from trajectory surface in spatiotemporal space", *Proc. CVPR'97*, pp. 200-205, 1997.

[10] M. Szummer and R. W. Picard, "Temporal texture modeling", in *Int. Conf. on Image Proc.*, Vol. 3, pp. 823-826, Sept. 1996.

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

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

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

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

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

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

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

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

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

[20] Adrian Low, *Introductory Computer Vision and Image Processing*, McGraw-Hill, 1991.

(a) Basketball (b) Flag

(c) Man (d) House

Fig.5. The HDTV video test sequences

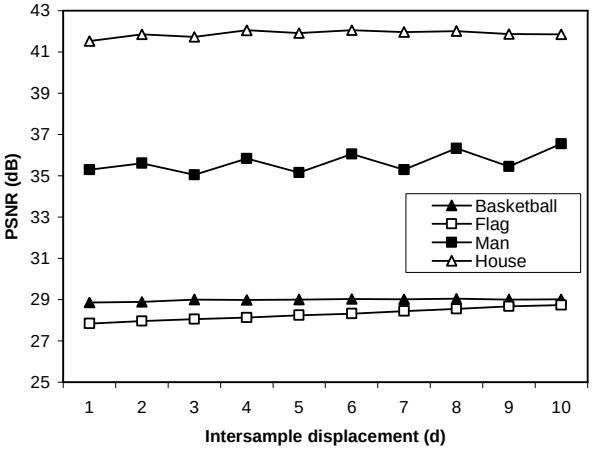

Fig. 6. Average PSNR of the motion compensated pictures for different values of the intersampling spacing distance (d)

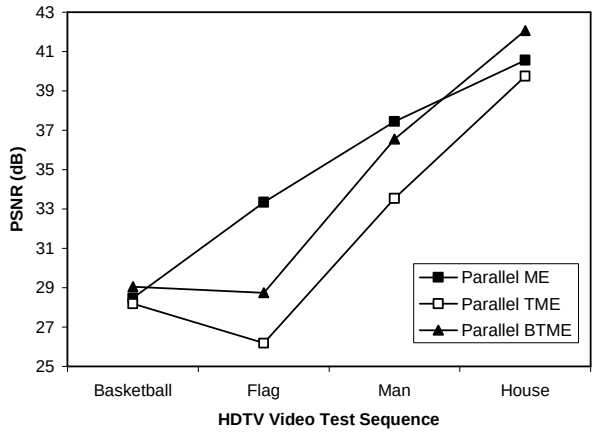

Fig. 7. The average PSNR for various sequence

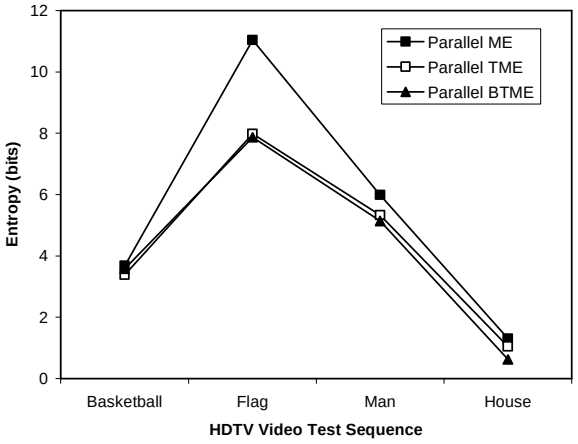

Fig. 8. The average Entropy for various sequence

Table 1. Speedup for various number of processors

Video Sequence	Parallel ME				Parallel TME				Parallel BTME			
	1	2	4	8	1	2	4	8	1	2	4	8
Basketball	1	1.96	3.95	7.90	1	1.97	3.89	6.97	1	1.92	3.80	7.37
Flag	1	1.99	3.94	7.77	1	1.98	3.72	7.22	1	1.96	3.68	7.15
Man	1	1.93	3.87	7.79	1	1.96	3.84	7.40	1	1.93	3.80	7.40
House	1	1.94	3.87	7.59	1	1.90	3.74	6.81	1	1.87	3.70	6.76

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

Video Sequence	Parallel ME				Parallel TME				Parallel BTME			
	1	2	4	8	1	2	4	8	1	2	4	8
Basketball	0.004	0.009	0.017	0.035	0.069	0.136	0.268	0.480	0.035	0.067	0.134	0.259
Flag	0.005	0.010	0.019	0.037	0.064	0.126	0.237	0.460	0.032	0.063	0.119	0.232
Man	0.005	0.009	0.017	0.035	0.058	0.113	0.221	0.426	0.029	0.056	0.110	0.215
House	0.004	0.009	0.017	0.034	0.063	0.121	0.237	0.432	0.032	0.060	0.119	0.259