Parallel Adaptive Temporal Prediction with Load Balancing for Fast Video Compression

S.Jeyakumar[1] *S.Sundaravadivelu* [2]

Abstract

Video image compression has been an area where the computational demand is far above the capacity of conventional sequential processing. In this paper, we present a parallel adaptive motion estimation model for video compression using cluster computing on a local network with balanced load. The method used for temporal prediction is adaptive, in the manner in which, frames with very few motion changes are predicted in its integer wavelet domain and for high motion activity frame, motion compensation is applied in its spatial domain. This approach gives good compression rate. Secondly we apply a parallel compression model by having a multiple networked heterogeneous personal computer systems that perform compression on different input frames simultaneously. Also computing load is distributed properly among all processors by resource management technique of cluster computing. The implementation result shows that the proposed parallel method has better speedup than sequential algorithm and very much suitable for online video applications.

Keywords : Data Parallelism, Task Parallelism, Motion Vector, Temporal Predictor, Message Passing Interface.

[1]Assistant Professor, Dr.Sivanthi Aditanar College of Engineering, Tiruchendur, India. Email : jkumar_simon@yahoo.co.in

[2]Professor, SSN College of Engineering, Chennai, India Email : sundaravadivelu@hotmail.com

1. Introduction

Video compression is a common need in today's multimedia and Internet world. The increased used of video data in telecommunications services, multimedia applications, the corporate environment, the entertainment industry, and at home has made digital video technology a necessity. Video images are moving pictures which are sampled at frequent intervals usually, 25 frames per second and stored as sequence of frames. A problem, however, is that digital video data rates are very large, typically in the range of 150 Mbits/sec. Data rates of this magnitude would consume a lot of the bandwidth, storage and computing resources in the typical personal computer. For this reason, video compression standards have been developed and intensive research is going on to develop effective techniques to eliminate picture redundancy, allowing video information to be transmitted and stored in a compact and efficient manner [6]. A video image consists of a time-ordered sequence of frames of still images. In video streams, adjacent frames tend to be very similar. MPEG compression makes use of this temporal redundancy of the data and allows video to be compressed using motion estimation. There are 2 kinds of frames defined in the video stream, each of which is compressed differently: Intra-frames (I-frames) and Inter-frames (P-frames). I-frames are treated as independent images, with no reference to any past frames. The encoding scheme used for I-frames is similar to JPEG LS compression,

where as P-frames are treated as predicted frames. An obvious solution to such frames would be predictive coding based on previous or future frames and the compression proceeds by coding the residual error [3],[8].

Though video compression system is available and several advanced compression techniques evolved, the compression tasks are computationally intensive, mainly due to the large amount of data to be processed and the time consuming repetitive operation of the processing algorithms. One of the ironies to come out of image compression research is that as the data rates come down where as the computational complexity of the algorithms increases. This leads to the problem of long execution times to compress an image sequence. It is apparent that, in order to transmit or store real time video, some scheme for fast video compression using parallel processing is necessary [9][10]. One of the significant characteristics of video compression algorithms, that make them very attractive for the use of parallel processing techniques, is that, the motion estimation algorithms can operate on different frames simultaneously, with each of these frames being coded separately. Hence frame coding facilitates making the image compression algorithm, adaptive to local image statistics and then be performed in parallel [13], [15].

The rest of the paper is organized as follows: A survey on the related work is presented in Section 2. Section 3, describes the proposed adaptive temporal prediction method. In Section 4, we present the parallel implementation model of proposed technique. Detailed experimental results and discussion have been given in Section 5, and finally, conclusions are drawn in Section 6.

2. Related Works

Video coding with good compression rate is useful for many real time applications, such as telemedicine, video conferencing and other multimedia systems [8].

Brunello *et al.* [2], introduced a temporal prediction technique based on block motion compensation and an optimal 3-dimensional linear prediction algorithm. In their scheme, based on motion information, the pixel to be coded is predicted by a linear combination of neighboring pixels in the current and reference frames. However, this method has more computation than reconstruction.

Ming *et al* [11] developed an adaptive combination of a spatial predictor and temporal predictor based on block motion compensation.

A wavelet-based lossless video coding algorithm is proposed by Gong *et al.* [7]. In this approach, block motion compensation was first performed in the spatial domain, and then wavelet coefficients of the prediction residuals were coded and transmitted.

In [14], Ying *et al.*, proposed an enhanced adaptive pixel based predictor which exploits the motion information among adjacent frames using extremely low side information. However, since the prediction is pixel based, the computational complexity is very high which may not be suitable for online real time applications.

In this paper, in order to improve the compression efficiency, we propose that, the motion vectors are adaptively estimated in spatial and wavelet domain based on inter frame similarity using correlation approach.

For improving the speed up of motion estimation, parallel processing is to be applied. Generally, approaches used for parallelism can largely be divided into two major

areas: architecture-driven approach and algorithm-driven approach [4],[10]. The first is the use of special purpose architectures designed specifically for performing operations in parallel, such as an array of DSP chips to implement JPEG and MPEG, high performance parallel computers and high speed networking. The second approach is algorithm driven, in which the structure of the compression algorithm is implemented with data and task parallelism [4]. In this paper we implement a parallel approach by distributing the work on a network of PCs.

3. Proposed Temporal Prediction

The objective of this work is to study the relationship between the operational domains for prediction, according to temporal redundancies between the sequences to be encoded. Based on the motion characteristics of the inter frames, the system will adaptively select the spatial domain or wavelet domain for prediction. The block diagram of proposed method is shown in Figure 1.

3.1 Adaptive Domain Selection

This step aims to determine the operational mode of video sequence compression according to its motion characteristics. The candidate operational modes are spatial domain and wavelet domain. The wavelet domain is extensively used for compression due to its excellent energy compaction. However, Gong *et al* [7] pointed out that motion estimation in the wavelet domain might be inefficient due to shift invariant properties of wavelet transform. Hence, it is unwise to predict all kinds of video sequences in the spatial domain alone or in the wavelet domain alone. Hence a method is introduced to determine the prediction mode of a video sequence adaptively according to its temporal redundancies. The amount of temporal redundancy is estimated by the inter frame correlation coefficients of the test video sequence [18].

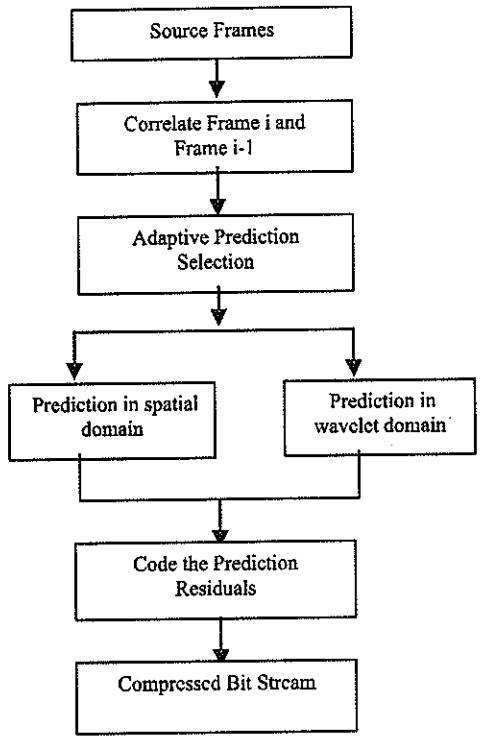

Figure 1 : Block Diagram Of Proposed Method

The inter frame correlation coefficient between frames can be calculated by (1). If the inter frame correlation coefficients are smaller than a predefined threshold, then the sequence is likely to be a high motion video sequence. In this case, motion compensation and coding the temporal prediction residuals in wavelet domain would be inefficient; therefore, it is wise to operate on the sequence in the spatial mode. Those sequences that have larger inter frame correlation coefficients are predicted in direct spatial domain. The frames that have more similarities with very few motion changes are coded using temporal prediction in integer wavelet domain.

$$C_{n,n+1} = \frac{\sum x \sum y \left(i_n(x,y) - \bar{i}_n\right) \cdot \left(i_{n+1}(x,y) - \bar{i}_{n+1}\right)}{\sqrt{\left(\sum x \sum y \left(i_n(x,y) - \bar{i}_n\right)^2 \cdot \sum x \sum y \left(i_{n+1}(x,y) - \bar{i}_{n+1}\right)^2\right)}} \quad (1)$$

3.2 Discrete Wavelet Transform

Discrete Wavelet Transform (DWT) is the most popular transform for image-based application [14]. A 2-dimensional wavelet transform is applied to the original image in order to decompose it into a series of filtered sub band images. At the top left of the image is a low-pass filtered version of the original and moving to the bottom right, each component contains progressively higher-frequency information that adds the detail of the image. It is clear that the higher-frequency components are relatively sparse, i.e., many of the coefficients in these components are zero or insignificant. The wavelet transform is thus an efficient way of decorrelating or concentrating the important information into a few significant coefficients. The wavelet transform is particularly effective for still image compression and has been adopted as part of the JPEG 2000 standard and for still image texture coding in the MPEG-4 standard. Figure 2 shows the representation of DWT sub bands of a three level multi resolution decomposition.

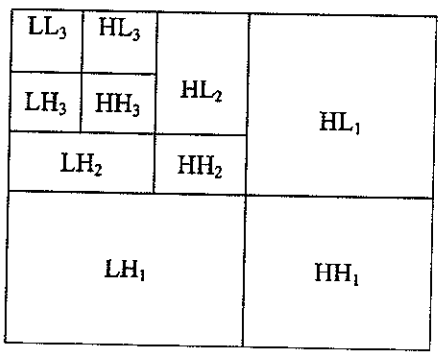

Figure 2 : DWT Sub Bands

The Haar wavelet is the first known wavelet and was proposed in 1909 by Alfréd Haar The Haar wavelet is the simplest possible wavelet with coefficients [0.707, 0.707]. The S transform is the integer version of the Harr transform [14] which has the lowest computational complexity, and reasonably well both for lossy and lossless compression. The forward S transform equations are given in (2).

$$h(i) = x(2i+1) - x(2i)$$

$$l(i) = x(2i) + \left\lfloor \frac{h(i)}{2} \right\rfloor \qquad (2)$$

where $x(i)$ is the input signal, $h(i)$ is the high frequency sub-band signal and $l(i)$ is the low-frequency sub-band signal.

3.3 Temporal Residual Prediction

Motion estimation obtains the motion information by finding the motion field between the reference frame and the current frame. It exploits temporal redundancy of video sequence, and, as a result, the required storage or transmission bandwidth is reduced by a factor of four. Block matching is one of the most popular and time-consuming methods of motion estimation [2],[3]. This method compares blocks of each frame with the blocks of its next frame to compute a motion vector for each block; therefore, the next frame can be generated using the current frame and the motion vectors for each block of the frame.

Block matching algorithm is one of the simplest motion estimation techniques that compare one block of the current frame with all of the blocks of the next frame to decide where the matching block is located [8]. Considering the number of computations that has to be done for each motion vector, each frame of the video is partitioned into search windows of size H*W pixels. Each search window is then divided into smaller macro blocks of size 8*8 or 16*16 pixels. To calculate the motion vectors, each block of the current frame must be compared to all of the blocks of the next frame with in the search range and the Mean Absolute Difference

(MAD) for each matching block is calculated using equation (1)

$$MAD_{m,n}(x,y) = \sum_{i=0}^{N-1} \sum_{n=0}^{N-1} [x(i,j) - y(i+m, j+n)] \quad (3)$$

where N*N is the block size, x(i,j) is the pixel values of current frame at (i,j) th position and y(i+m,j+n) is the pixel value of reference frame at (i+m,j+n) th position. The target mapping of current frame and reference with in the search range is shown in figure 3.

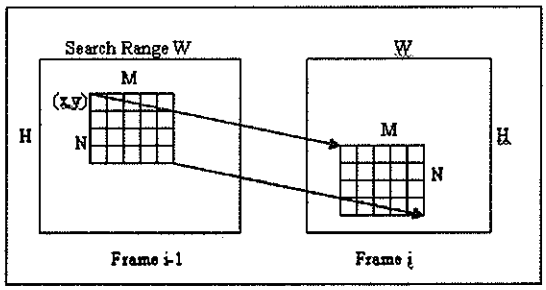

Figure 3 : Inter Frame Target Window Mapping

The block with the minimum value of the Mean Absolute Difference (MAD) is the preferred matching block. The location of that block is the motion displacement vector for that block in current frame. The best motion vector for the target window with the minimum MAD is determined by,

$$(m_0, n_0) = \{\text{minimum MAD } (T_w)\} \quad (4)$$

where (m_o, n_o) indicates the motion displacement of the target window Tw. Then the temporal predictor of pixel $p_i(x,y)$ can be obtained by,

$$\hat{p}^T(x,y) = p_{i-1}(x+m_0, y+n_0) \quad (5)$$

and the temporal prediction residual is

$$e_2 = p_i(x,y) - \hat{p}_i^s(x,y) \quad (6)$$

3.4 Coding the Prediction Residual

The temporal prediction residuals from adaptive prediction are encoded using Huffman codes. Huffman codes are used for data compression that will use a variable length code instead of a fixed length code, with fewer bits to store the common characters, and more bits to store the rare characters. The idea is that the frequently occurring symbols are assigned short codes and symbols with less frequency are coded using more bits. The Huffman code can be constructed using a tree. The probability of each intensity level is computed and a column of intensity level with descending probabilities is created. The intensities of this column constitute the levels of Huffman code tree. At each step the two tree nodes having minimal probabilities are connected to form an intermediate node. The probability assigned to this node is the sum of probabilities of the two branches. The procedure is repeated until all branches are used and the probability sum is 1.Each edge in the binary tree, represents either 0 or 1, and each leaf corresponds to the sequence of 0s and 1s traversed to reach a particular code. Since no prefix is shared, all legal codes are at the leaves, and decoding a string means following edges, according to the sequence of 0s and 1s in the string, until a leaf is reached.

The code words are constructed by traversing the tree from root to its leaves. At each level 0 is assigned to the top branch and 1 to the bottom branch. This procedure is repeated until all the tree leaves are reached. Each leaf corresponds to a unique intensity level. The codeword for each intensity level consists of 0s and 1s that exist in the path from the root to the specific leaf.

4. Implementation Of Parallel Model

In this paper, we introduce parallel image processing approach that applies distributed client-server computing concept. This technique uses the power of local computer network with master-slave concept [5],[9],[12]. The environment consists of a server with a number of workstations. A simple master-slave computing model is shown in the figure 4.

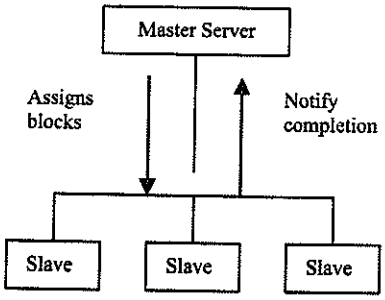

Figure 4 : Master-Slave Model

This is a simple approach which distributes independent, non-overlapping image blocks on a multi, single processor cluster of workstations, using Message Passing Interface mechanism. The master-slave concept is the standard approach, in which the master sends the data to a slave and the slave sends back the output after computation [13]. The system consists of a master server and many slave processors. The master server controls synchronization of all processes, assigns slave processes which blocks to process, notifies the output server (same machine or another client) which blocks are done. The output server combines output blocks to a output file and slave processes perform processing and outputs results, notifies master process which and when the assigned blocks are done.

4.1 Parallel Temporal Prediction

The block diagram of the proposed parallel implementation of motion vector estimation and motion vector approximation is shown in figure 6.

The algorithm and steps for our parallel work is as below.

a. Maintain a collaborative memory in the master server to keep the video sequences as a queue [5].

b. By default, the first and last frames are considered as key frames and compressed using spatial predictor method before the parallel run.

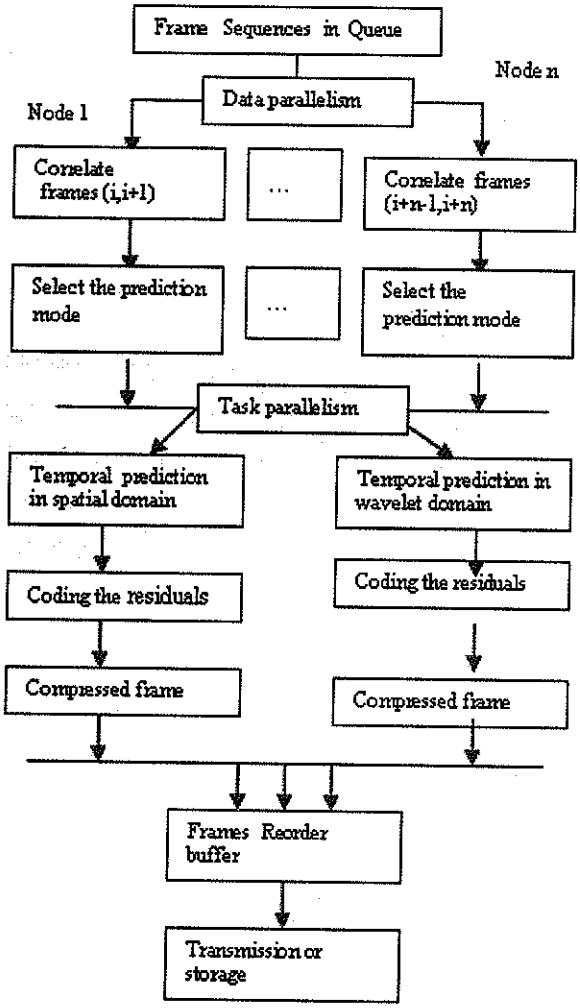

Figure 6 : Parallel Motion Estimation Model

c. Server then distributes the frames among all the processors in the cluster of workstations sequentially.

d. Each slave then proceeds through its assigned frame, performs the temporal prediction adaptively.

e. After finishing up, the slave sends message to master server and it writes 1 to the completion register of the respective frames.

f. When the completion register of specific workstation becomes 1, the processed frame is sent to the output server from the slave processor.

g. The master server computes the next chunk of frames for this slave to compress, encode a message and send it to the slave.

h. The process is repeated and the server will remain in the waiting state till the completion of all slave processes.

4.2 Load Balancing

The key to good parallelization is, how to synchronize the slave processes, so that no slave process is kept waiting for assignment from the master to get frames to compress and that output server is not waiting for a slower slave process, for the output frames to generate the compressed file.

Load balancing is the major criterion to improve throughput or speed up execution of the set of jobs while maintaining high processor utilization. Load balancing is the allocation of the workload among a set of co-operating nodes [1],[6],[15]. Generally, when working with heterogeneous nodes in a cluster, the computing powers of the intervening nodes are a factor used to analyze the distribution of the work to be done. In task parallelism, if the type of work is static, a predictive load balancing function can be used. It is defined as

$$D = F(P_i, T_L) \qquad (7)$$

where D is the distribution function, P_i is the computing power of processor *i* and is T_L the total work. The total workload T_L will be allocated to the N processor at the moment of starting the application, according to the distribution function D. In parallel motion estimation for video compression, a group of frames is assigned to a particular node depending on the computing power available for that node. The typical computing resources consist of CPU Clock speed in MHz and memory capacity in Megabyte (MB) or Gigabyte (GB). This speed and memory characteristics then can be used to distribute the computing load evenly. Hence, for data distribution, the number of frames assigned to a node is related to the computing power of that node relative with the total computing power of nodes. This can be expressed as:

$$Nf_i = \frac{(Ws + Wm)_i}{\sum (Ws + Wm)_j} Tf \qquad i, j = 1,\ldots,n \qquad (8)$$

where Nf_i, Ws_i, Wm_i and Tf are the number of frames assigned to node i, weight of node i according to its speed and memory capacity and total number of frames respectively. With weighted round robin, the loads are distributed among the participating nodes on a round robin fashion. The capacity information of each node is collected and sent to master node by the resource-monitoring system before start of assigning frames. During each frame processing the master node distributes the frame accordingly, and this process will continue until all frames are processed.

5. Results And Discussion

The proposed compression algorithm has been implemented in Java platform for standard test video sequences. In order to check the efficiency of our proposed adaptive temporal prediction and its parallel implementation, 50 video frames (Tennis sequences) were processed with 4 slaves and one master server. By default, first and last frames are considered as I-frames and remaining 48 frames are compressed using block based temporal prediction. The parallel output is verified with sequential processing and in all cases the output matches with the output produced by sequential version.

The experiments were done on a cluster architecture consisting of 1 server 4 nodes:

- Server - HP Proliant Server MI 350
- Slave #1 - Pentium IV 2.66 GHz 512 MB RAM

- Slave #2 - Pentium Quad Core 2.4 GHz 1 GB RAM
- Slave #3 - Pentium D CPU 2.8 GHz 480 MB RAM.
- Slave #4 - Pentium IV 2.4 GHz of 256 MB RAM

The systems are interconnected to one 24-port 10/100 mbps switch using CAT 5E UTP cable. The prediction results for the Tennis sequence frames 1 and 3 are shown in figures 7 and 8 respectively. Frame 1 has high motion characteristics from frame 0 and hence its motion vectors are predicted and residuals are compressed in spatial domain. Frame 3 more similarity with frame2 and its motion estimation is done in wavelet domain.

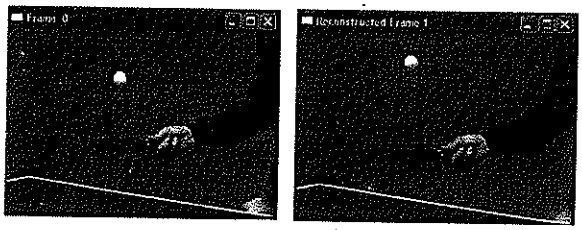

Figure 7 : Prediction for Tennis Frame 1 in Spatial Domain

(a. Frame 0 b. Reconstructed Frame 1)

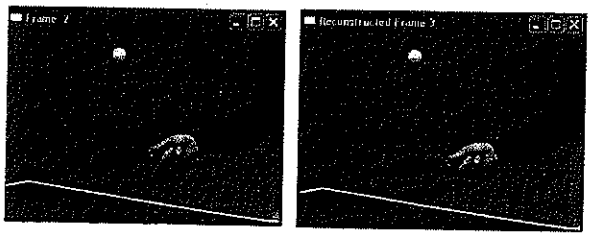

Figure 8 : Prediction for Tennis Frame 3 in Wavelet Domain

(a. Frame 2 b. Reconstructed Frame 3)

5.1 Compression Ratio

To analyze the results of our proposed adaptive prediction, Compression Ratio (CR) and Peak Signal to Noise Ratio (PSNR) parameters are used. Compression Ratio (CR) is defined as the ratio between the number of bits required to store the image before compression (I) and the number of bits required to store the image after compression (O).

Table 1 lists the motion characteristics of each frame over its previous frame, its prediction mode and compression ratio for the Tennis frame sequences (1-9). The prediction threshold is fixed as 0.99. The frames with correlation coefficient 0.99 and above are considered as having, low motion features and their prediction domain is wavelet. The Frames that have correlation coefficient below 0.99 are high motion frames and hence their prediction is direct spatial domain.

Table 1 : Compression Ratio For Test Video Sequence (Tennis Frames 1-9)

Frame	Correlation coefficient	Prediction domain	Compression ratio
1	0.983613	Spatial	6.86
2	0.985742	Spatial	6.73
3	0.994431	Wavelet	7.91
4	0.997359	Wavelet	7.25
5	0.995662	Wavelet	7.93
6	0.992310	Wavelet	7.96
7	0.971542	Spatial	6.85
8	0.981276	Spatial	6.62
9	0.975463	Spatial	6.37

From the results of table 1 it is obvious that, temporal prediction in wavelet domain has high compression ratio than spatial prediction. However, spatial prediction is faster and motion estimation in the wavelet domain is inefficient for high motion frames, due to shift invariant properties of wavelet transform.

5.2 Peak Signal to Noise Ratio

To analyze the quality of proposed adaptive prediction method, the Peak Signal to Noise Ratio (PSNR) is calculated between the original frame and reconstructed frames by,

$$PSNR = 10\ Log_{10}\left(\frac{255^2}{mse}\right) \quad (9)$$

where, Mean Square Error *(mse)* is

$$mse = \frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n}\left(y_{i,j} - x_{i,j}\right)^2 \quad (10)$$

In (10), m and n denote respective number of rows and columns in the image, *mse* is the calculated Mean Square Error, $y_{i,j}$ is decompressed image at location (i,j) and $x_{i,j}$ is original image at location (i, j). Table 2 shows the possible combination of spatial-wavelet domains for motion estimation in case of low motion frames and frames with high motion characteristics.

Table 2 : Combination of Domains For Prediction

Possible Combination	Low motion frame	High motion frame
C1	Spatial	Spatial
C2	Wavelet	Wavelet
C3	Wavelet	Spatial

Table 3 : PSNR Value Of Various Samples Of Adaptive Temporal Prediction Tennis

Frame	C1	C2	Proposed C3
1	33.34	32.73	33.34
2	33.38	33.25	33.38
3	32.16	33.85	33.85
4	32.08	32.84	32.84
5	31.97	33.26	33.26
6	32.16	33.12	33.12
7	33.05	32.34	33.05
8	33.26	32.72	33.26
9	33.49	32.81	33.49

Table 3 gives the performance comparison of the quality parameter in terms of PSNR for the proposed adaptive method with the existing spatial alone and wavelet alone approaches. The PSNR values in C3 column of our various samples of input in Table 3, shows that the temporal prediction using the proposed adaptive method is robust and it can be well adopted for lossless video sequence compression.

5.3 Speed Up of Parallel Implementation

The table 4 shows the execution time, of a cluster of 4 workstations and one master server and Table 5 is the speedup of parallel implementation of the temporal predictor. In this paper, only the calculation power of each processor has been taken into account and the communication cost due to message passing is not considered.

Table 4 : Fixed Frames Distribution For Each Slave And Its Execution Time (In Seconds)

Slave No.	Frames Processed	Total Time	Average Time	Frame Rate
Slave #1	12	763.44	63.62	0.94
Slave #2	12	565.68	47.14	1.27
Slave #3	12	604.20	50.35	1.19
Slave #4	12	829.44	69.12	0.86

The speed up for the proposed parallel method is calculated by

$$Speedup = \frac{Execution - time_{Sequential}}{Execution - time_{Parallel}} \quad (11)$$

The total time taken by a single PC with the said configurations to complete the motion vector estimation and approximation for 48 frames and the maximum time consumed by parallelization of the same operation are listed in Table 5.

Table 5: Speedup of Parallel Implementation

Slave Number	Sequential time (Sec)	Parallel time (Sec)	Speed up
Slave #1	2943.36	829.44	3.54
Slave #2	2202.72	829.44	2.65
Slave #3	2357.46	829.44	2.84
Slave #4	3232.80	829.44	3.89

The total time taken by a single PC (slave #3 without communication overhead) to complete the motion vector estimation and approximation for 48 frames is 3053.76 seconds and the time consumed by parallelization the same operation with 2 to 4 slaves and speed up are listed in Table 6.

Table 6 : Speedup Of Parallel Implementation

No. of Slaves	Total time in Sec	Speed-up
2	1706.71	1.79
3	1131.02	2.70
4	850.62	3.59

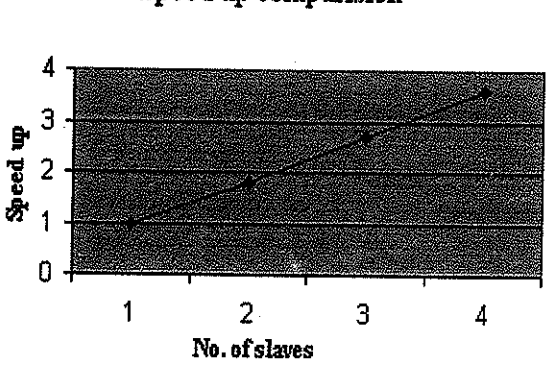

Figure 9 : Speed Up Comparison

Figure 9 shows the speed up curve. We observe from the figure that the speedup for parallel video compression is nearly linear with the number of processors. However, for very large number of processors in the cluster, the overall execution time increases, because of communication overhead. Hence, if communication overhead becomes greater than the computing time, the number of processing nodes in the cluster is too limited.

5.4 Results of Load Balancing

When working with cluster of heterogeneous nodes, the load balancing of an application has a direct impact on the speedup to be achieved as well as in the performance of the parallel system. Hence, predictive load balancing formula as we have defined in (8) is applied to calculate the number of frames to be processed by each node. The data of table 1 shows the frames processed by each slave produced by the algorithm for balanced load distributions.

Table 6 : Frames Distribution For Each Slave And Execution Time (In Seconds) As Per Balanced Load Algorithm

Slave No.	Frames Processed	Total Time	Average Time	Frame Rate
Slave #1	11	578.82	52.62	1.14
Slave #2	15	714.60	47.64	1.24
Slave #3	13	674.18	51.86	1.15
Slave #4	9	541.71	60.19	1.00

Figure 10 shows the computational load distribution of parallel adaptive temporal prediction algorithm for Tennis sequence. The parallel execution time will be greatly affected by the longest processing time among processors where the workload is high. To cope up with this behavior, the computational load is balanced all over the nodes according to its computing power.

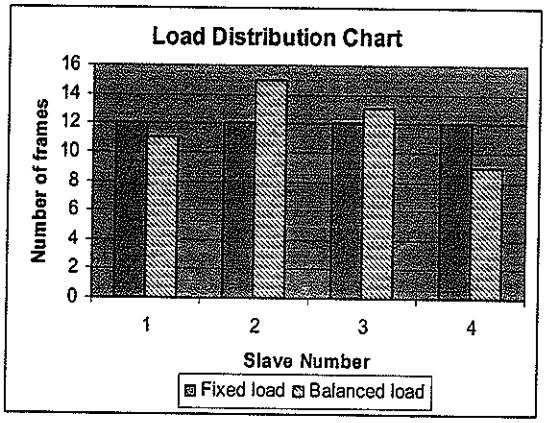

Figure 10 : Balanced Load Distributions For Each Slave

The speedup for the proposed load balanced parallel method is calculated by

Throughput = Execution time $_{fixed}$ / Execution time $_{Balanced}$

Table 7 : Speedup Of Parallel Implementation with Balanced Load

Total time Fixed load	Total time Balanced load	Throughput
2762.76	2509.31	1.10

Table 7, shows the computing cost of fixed and balanced load and figure gives the comparison of computing cost of fixed load and balanced load frame allocation among slaves.

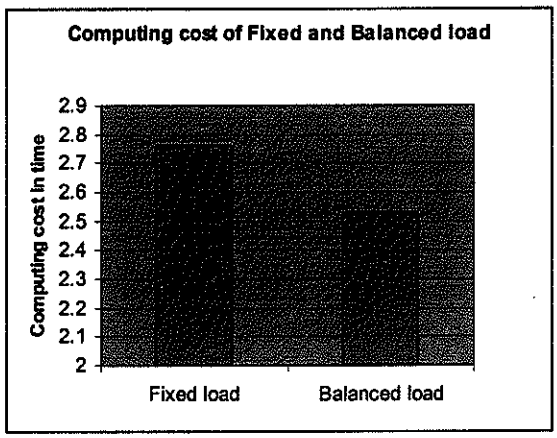

Figure 11: Comparison of Computing Cost Of Fixed and Balanced Load

From the results obtained, it is observed that, the speedup achieved with the load balancing is slightly increases than fixed load method. This parallel implementation of adaptive temporal prediction for video compression with load balancing on a cluster environment of 1 master and 4 heterogeneous slaves is 1.10 times faster than static load distribution. In real world cluster environment, as the number of slaves increases the speed up will also gets increased, and so notable computing time could be saved with balanced load.

However, Communication between two processing steps is expensive in computing time. This is due to the large amount of regular communications required by message passing interface techniques [4],[10]. Our future work will concentrate on optimal data organization for reducing the communication overhead and developing a component based, distributed motion estimation method for web applications.

6. Conclusion

In this paper, we have implemented a parallel temporal prediction model for fast video sequence compression. The motion estimation is done adaptively in spatial domain or wavelet domain, according to its motion features. Hence a good compression rate is achieved for video compression than the existing single domain prediction methods. Further, parallel execution is implemented using client-server architecture on a cluster of local work stations. It is a costless method than other parallel schemes, since computer network is common every where. The speed up comparison shows that, the execution time of video compression can be much reduced by performing the compression of different frames simultaneously, on multiple PCs of a cluster environment. Moreover, an unbalanced workload across the processors can significantly reduces the performance of the parallel program. In order to overcome this issue, a balanced load distribution algorithm based on computing power of each node is introduced. This improvement enhances the speed up and has better behavior than direct fixed load frame distribution. From the experiments conducted and results obtained, this method is found to be very useful for real time on-line applications such as telemedicine, video conferencing and video surveillance system.

References

[1] Armando.E, Marcelo.R, Naiouf, Laura.C, *"Dynamic Load Balancing in Parallel Processing on Non-Homogeneous Clusters"*, Journal of Computer Science, Vol. 5, No. 4, December 2005.

[2] D. Brunello, G. Calvagno, G. A. Mian and R. Rinaldo, *"Lossless compression of video using temporal information"*, IEEE Transaction on.

Image Processing, Vol. 12, No. 2, PP. 132–139, Feb. 2003.

[3] E. S. G. Carotti, J. C. De Martin and A. R. Meo, *"Low complexity lossless video coding via spatio-temporal prediction"*, in Proc. Int. Conf. Image Processing, Vol. 2, PP. 197–200, Sep. 2003.

[4] Christiana Nicoles and Peter Jonker, *"A data and task parallel image processing environment"*, Journal of Parallel computing, Elsevier Science Publishers, PP. 945 – 965, 2002.

[5] Clematis. D. D.Agostino and A. Galizia, *"A Parallel IMAGE Processing Server for Distributed Applications, Parallel Computing: Current & Future Issues of High-End Computing"*, Proceedings of the International Conference ParCo 2005.

[6] Donaldson.V, Berman.F, Paturi.R, *"Program Speedup in a Heterogeneous Computing Network"*, Journal of Parallel and Distributed Computing 21:3 (6/1994), 316-322.

[7] Y. Gong, S. Pullalarevu and S. Sheikh, *"A wavelet-based lossless video coding scheme"*, in Proc. Int. Conf. Signal Processing, PP. 1123–1126, 2004.

[8] Iain.E.G. Richardson, *"H.264 and MPEG-4 Video Compression"*, John Wiley & Sons, September 2003.

[9] Kambiz Tavassoli and Wael Badawy, *"A Prototype for Parallel Motion Estimation Matching Algorithm Architecture Using Full-Search Block"*, IEEE International Workshop on Digital and Computational Video, Nov' 2002.

[10] Ke Shen, Gregory W. Cook, Leah H. Jamieson and Edward J. Delp, *"An Overview of Parallel Processing Approaches to Image and Video Compression"*, SPIE, 2000.

[11] Z. Ming-Feng, H. Jia and Z. Li-Ming, *"Lossless video compression using combination of temporal and spatial prediction"*, In Proc. IEEE. Int. Conf. Neural Networks Signal Processing,PP.1193–1196, Dec. 2003.

[12] Soo-Young Lee and J. K. Aggarwal, *"A System Design and Scheduling Strategy for Parallel Image Processing"*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 12. No. 2, PP. 193-204, February 1990.

[13] Teddy Surya Gunawan and Cai Wen Tong, *"Parallel Motion Estimation on SMP System and Cluster of SMPs"*, IEEE Proceedings of the International Parallel and Distributed Processing Symposium, 2002.

[14] Ying Li and Khalid Sayood, *"Lossless Video Sequence Compression Using Adaptive Prediction"*, IEEE Transaction on Image Processing, Vol. 16, No.4, Apr.2007.

[15] Yunsong Wu and Graham Megson, *"Parallel Linear Hash table Motion Estimation Algorithm for Parallel Video Processing"*, IEEE Proceedings of the International Symposium on Parallel Computing in Electrical Engineering, 2006.

Author's Biography

S.Jeyakumar is working as Assistant Professor in Information Technology at Dr.Sivanthi Aditanar College of Engineering, Tiruchendur, India. He has completed BE degree in CSE and MTech in Information Technology. He is presently doing PhD programme in Anna University, Chennai. His research area includes video compression, optical image processing and parallel image processing.

Dr.S.Sundaravadivelu is presently working as Professor in ECE at SSN College of Engineering, Chennai, India. Earlier, he worked at Thiagarajar College of Engineering, Madurai as Head of ECE department. He completed his ME degree from Regional Engineering College, Trichy and awarded PhD by Madurai Kamaraj University. He has over 25 years of teaching and research experience. He is guiding several research scholars. His research interest includes optical signal processing, digital image processing and parallel computing.