A Parallel Algorithm for Discrete Gabor Transforms

Kshitij Sudan
Department of Computer Engineering,
Delhi College of Engineering
University of Delhi, India.

Nipun Saggar
Department of Information Technology,
Delhi College of Engineering,
University of Delhi, India

Asok De
Department of Electronics and Communication Engineering,
University of Delhi ,India

Abstract - *Serial algorithms to evaluate the Gabor transform of a discrete signal are bound by the length of signal for which the transform can be evaluated. The time taken, if machine memory and other factors are ignored, grows as order $O(N^2)$ making it unsuitable for transforms of large signal lengths. In this paper, we present a parallel algorithm to generate the computationally intensive Gabor transforms of a discrete signal. This algorithm is suited to both shared memory and message passing systems and is independent of the underlying hardware. The performance was tested on a Linux cluster build on the message passing mode land linear speedup was observed. The network overheads are measured and the algorithm performance is proven to be independent of network contention for practical cases, although it provides a theoretical bound on achievable speedup. The design of the algorithm parallelizes the most computationally intensive sub-process of transform evaluation and also makes it suitable for scaling to larger systems.*

Keywords: Parallel algorithms, distributed memory parallel algorithms, high performance computing, Gabor transforms, non-orthogonal series expansions.

1 Introduction

Gabor transforms [1], owing to their excellent localization
In time and frequency domains, find an increasing number of applications in signal processing and mathematical analysis.

These transforms are analogous to Fourier transforms except that they localize the signal in both frequency and time domain. Gabor expansion of signal refers to expansion of an arbitrary signal into a series of elementary terms which are time-frequency shifted copies of a given "Gabor atom" [1]. It is well-known that the main difficulty involved in the Gabor expansion is to select a suitable set of Gabor coefficients [2]. Several algorithms and analytical methods describe the procedure to evaluate the coefficients [3], but none of them is practically able to produce results with extreme accuracy in small time durations, especially when the discrete signal length is very large. In finding suitable Gabor coefficients, the most computationally intensive part is considered to be the evaluation of so-called dual Gabor atom [4].

The fact that dual atom can be evaluated efficiently using the properties of Gabor-matrix structure has been exploited to yield efficient serial algorithms [11]. However, these algorithms are still bound by the machine memory and computational resources and thus are unable to process large signals.

In the present paper, we exploit the structure of Gabor matrix [13] to distribute the task of evaluation of individual elements of itself over a parallel computer. This new approach not only allows evaluation of dual atoms for large signal lengths but scales nicely with increasing number of compute cores of the system. It can thus be seen as continuing the advancement in serial algorithms with added capability of allowing large discrete signals.

The paper is organized into five sections. Section 2 describes a serial algorithm which exploits the block and banded structure of Gabor matrix. In section 3 we present the parallel algorithm and explain its operation and expected speedup and scalability. In section 4 numerical results are presented showing the actual speedup obtained with large signal lengths. The paper is concluded in section V with remarks toward future work in the area.

2. Serial algorithms

We refer to [5], [7], [9], [10] and [12] for basic explanation of the algorithms for evaluation of Gabor coefficients. Results directly employed in parallel algorithm are reproduced here for convenience however.

For signals represented as vectors, we define a signal of length N to be a vector of N discrete values. Thus a signal can be represented as N-*periodic* sequences over integers $\mathbf{Z}$; therefore it can be represented as,

$$x = \{x_1, x_2, \quad \ldots x_N\} \qquad (2.1)$$

Due to the periodic property of these sequences one can write,

$$x(j + kN) = x(j) \text{ , for } j, k \in Z \qquad (2.2)$$

Two operators required to build the Gabor matrix G [13], *rot()* and *rotm()* are defined next. The *rot(. , a)* operator is defined as:

$$rot(h, m) = (h_{N-m+1}, h_{N-m+2}, \ldots, h_N, h_1, h_2, \ldots, h_{N-m}) \qquad (2.3)$$

where h is a signal of length N and m is the rotation number. The *rotm(. , a)* operator is defined as:

$$rotm(M, a) = (rot(M_1, a), rot(M_2, a), \ldots, rot(M_n, a)) \qquad (2.4)$$

where M is a $m \times n$ matrix and M_k is the *k-th* column of M with a being the rotation number.

It is noted here that the *rot(. , a)* operator corresponds to ordinary time translation of periodic signals and hereafter also referred to by the symbol T_a.

Using these notations, the Gabor matrix *G,* for a given signal *g* of length *N* and lattice constants *a* and *b* [2], [12], can be constructed by evaluating each individual entry of the matrix by the following relation [7]:

$$G_{k+1,j+1} = \begin{cases} \tilde{b}\sum_{n=0}^{\tilde{a}-1} T_{na}\,\overline{g(k)}\,T_{na}\,g(j) & if\,|k-j|\ is\ divided\ by\ \tilde{b} \\ 0 & otherwise \end{cases} \quad (2.5)$$

where $k, j = 0,1, \ldots, N-1$ and $\tilde{b} = \frac{N}{b}, \tilde{a} = \frac{N}{a}$

It can be observed from this equation, since *G* is a $N \times N$ matrix, the complexity of the algorithm is O(N^2).

Qui describes an algorithm for evaluation of *G* matrix using its block-banded structure in [6], [8] and [13]. It involves first building a $N \times a$ matrix *B* and subsequently composing the *G* matrix from proper rotations of *B* matrix affected by using the *rotm()* operator. The general entries of *B* matrix are given by the following relation:

$$B_{k+1,j+1} = \tilde{b}\sum_{n=0}^{\tilde{a}-1} T_{na}\,\overline{g(k)}\,T_{na}\,g(j) \quad (2.6)$$

where $j = 0,1,\ldots,a-1$ and $k = p.\tilde{b} + j$ for $p = 0,1,\ldots,b-1$

Using this, the ***G*** matrix can be written as,

$$G = [B,\ rotm(B,a),\ \ \ldots,\ rotm(B,(\tilde{a}-1)a)] \quad (2.7)$$

This procedure reduces the complexity from O(N^2) to O(N) as now setting up of the *G* matrix requires only $ba + Nb$ multiplications and some rotations [11]. At this stage it's relevant to note that although complexity of the algorithm is reduced to linear, it however is still not suitable for computing *G* matrix for large signal length. Depending on the machine memory and computational capabilities, this large can vary from N=200, a=10 and b=5 for a low end desktop machine with 128MB main memory and a popular processor running at 1.6Ghz, to N=4000, a=100 and b=20 on a high end desktop with 512MB memory and a dual core processor running at 3.0Ghz.

3. The algorithm

The major bounding constraint in computation of *G* matrix for large signal lengths is machine memory, as computations can take sufficiently large time duration to complete. The proposed algorithm has a major advantage in this respect since it deals with distributing blocks of computations and thus eases the constraint on machine memory. The advantage of using multiple processing units reflects itself in almost linear reduction in required time to compose the full *G* matrix.

The algorithm works by distributing the generation of Gabor matrix *G* to different compute nodes. Our parallelization of the method proposed by Qiu [7], [13] proceeds by first building the $N \times a$ matrix *B* serially, followed by parallel rotation of *B* by varying amounts of rotation constant, in the sense defined by a matrix rotation operator *rotm(),* to yield the building blocks of Gabor matrix.

For a *k* processing unit machine, the first

$$p = \left\lfloor \frac{\tilde{a}}{k} \right\rfloor \ , where\ \tilde{a} = \frac{N}{a} \quad (3.1)$$

sub-matrices are mapped to first $(k-1)$ units and the k^{th} unit shall compute

$$q = \tilde{a} - (k-1)\left\lfloor \frac{\tilde{a}}{k} \right\rfloor \quad (3.2)$$

independent $N \times a$ matrices.

We note here that such a procedure is possible only because of block and banded structure of the Gabor matrix [13], [14]. The calculation of the dual atom [12] can now be done using the relation:

$$\tilde{g} * G = g \quad (3.3)$$

where $\tilde{g}$ is the dual [4] of the discrete signal *g* whose length we have mentioned earlier as *N*. We use standard parallel routines from ScaLAPACK [15] library to evaluate the dual using Eq. 3.3. We now give a formal listing of the parallel algorithm.

Algorithm: Computing dual of a given Gabor atom

Step 1: Build the $B_{N\times a}$ matrix on the head node serially.

Step 2: Pack and broadcast the *B* matrix to all the compute nodes.

Step 3: DO PARALLEL

Apply *rotm()* operator on *B* matrix with amount of rotation specific to each compute node.

END PARALLEL

Step 4: Communicate back the new rotated matrices to head node, which builds the final Gabor matrix *G* by concatenation of properly rotated *B* matrices.

Step 5: Calculate the inverse of Gabor matrix *G*, using standard parallel routines provided in ScaLAPACK.

Step 6: Using ScaLAPACK routines, find the dual Gabor atom by matrix multiplication of Gabor atom (represented as a $1 \times N$ matrix) with the Gabor matrix ($N \times N$).

The specific issue of network contention is discussed in the following section.

4. Numerical results

In this section we present numerical results obtained from experiments carried out on a 10 node Linux cluster of Intel desktops interconnected by a standard 100MBps Ethernet LAN.

4.1 Speed-up

The observed speed-up over the serial algorithm is almost linear as shown in Fig.1 and Fig.2 for differing signal lengths. In these figures we plotted the time taken to compose the *G* matrix after the *B* matrix has been transmitted to all the compute nodes. As can bee seen from the plots, the expected

and experimental times are in excellent agreement thus validating the fact that linear speedup's are achieved by the algorithm. These linear speedup's are expected as the computations are mutually independent and network contention is a small fraction of computation time. We discuss this issue of network contention shortly.

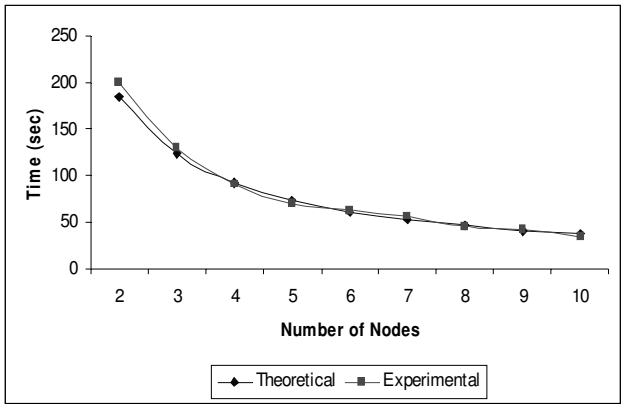

Fig. 1 Time vs. Number of Nodes for Signal length N=6500.(Theoretical and Experimental)

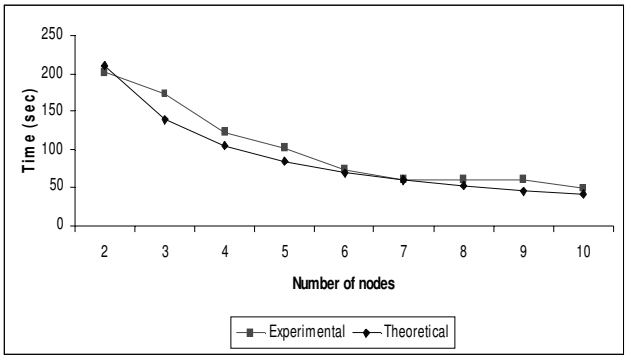

Fig. 2 Time vs. Number of Nodes for Signal length N=7222.(Theoretical and Experimental)

4.2 Scalability

The algorithm scales easily with increasing number of compute cores. It can be observed that the only parameter which needs adjustment for varying number of compute cores was the number of independent $N \times a$ matrices p and q, which a core is assigned to compute. This is a trivial adjustment as the tuple <p,q> is completely and precisely defined by Eq. 3.1 and Eq. 3.2.

4.3 Network Contention

The amount of data transferred from a compute node varies depending on the signal length. For N=6500, a=65 and b=50 the amount of data transfer is plotted as a function of number of compute nodes of the Linux cluster in Fig. 3. This graph is expected as each node gets smaller share of total workload while the number of nodes is increased thus producing lesser data to be transmitted. The amount of time taken to transfer this data for a standard 100MBps Ethernet LAN is plotted in Fig. 4 which shows that with increase in number of nodes the network contention does not increase. In fact it stays constant for this particular data set range as arbitration between sender and receiver takes larger time than that required for data transfer.

The reason for such a low impact of network resources on the algorithm is the fact that computational times are several orders of magnitude larger than network transfer times. In the experiments with N=6500, a=100 and b=65, it was observed that when all 10 nodes are used, the computational time is 90,000 times larger than data transfer time and while using only 2 nodes it was nearly 500,000 times.

It can thus be deduced that network contention is not a major issue in performance of this algorithm, especially when computation times are several orders of magnitude larger. A point of caution here is due however. If the number of compute cores is increased to a very large value (say 900,000) for relatively small signal length (say N=6500) then the network transfer time is the bounding factor for the computational time. This can be explained considering the fact that in such a case the computational time has reduced to such a small value due to linear speedup that now network arbitration time, between the sender and the receiver, (as the amount of data to be transferred also has reduced linearly) is the upper limit for evaluation of results. This represents the theoretical limit of speedup that can be achieved with the algorithm.

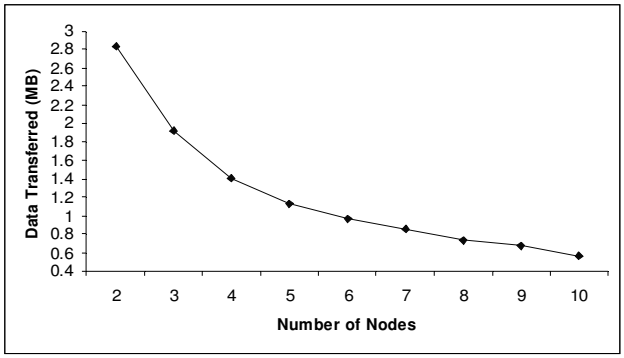

Fig. 3 Data Transferred by Each Node vs. Number of Nodes

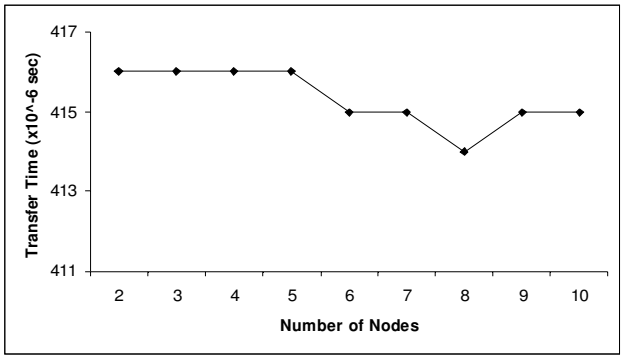

Fig. 4 Data transfer time between a pair of node vs. Number of nodes

5. Conclusion

Building on previously described serial algorithms; we have developed a parallel algorithm capable of evaluating the computationally intensive portions of Gabor transform

efficiently and with nearly linear speed-up. The algorithm shows no degradation with increasing number of computing units thus addressing the specific issue of scalability. Network contention, though it presents the theoretical limit to the achievable speedup, is negligible in practical situations owing to difference of several orders of magnitude between computational time and data transfer time over the network. The algorithm thus performs well on all the three major metrics faced by parallel algorithms.

The future direction of work on this algorithm would involve a dynamic allocation of compute cores based on parameters like signal length, which gives the best cost vs. performance ratio. A flat model of communication, which is currently adopted for the algorithm, will be modified to introduce a hierarchical or a peer-to-peer model to aggregate data as network transfers are cheaper than computations on the processing cores.

6. Acknowledgement

The authors wish to thank and extend their appreciation for Mr. N. S. Raghava for constant support during the course of this work.

7. References

[1] D. Gabor, "Theory of communication." *J. IEE (London)*, 93(III):429-457, November 1946.

[2] M.J. Bastiaans, "Gabor's Expansion of a Signal into Gaussian Elementary Signals." *Proceedings of the IEEE*, 68:538--539, April 1980.

[3] M.J. Bastiaans and M.C.W. Geilen, "On the discrete Gabor transform and the discrete Zak transform", *Signal Proc.*, 49(3):151--166, 1996.

[4] T. Werther, Y.C. Eldar, and N.K. Subbana., "Dual Gabor frames: Theory and computational aspects", *IEEE Trans. Signal Processing*, 53(11), 2005.

[5] P.Prinz, *Theory and algorithms for discrete 1-dimensional Gabor frames*, Masters Thesis, University of Vienna, Austria, 1996.

[6] H.G. Feichtinger and O. Christensen, "New efficient methods for Gabor analysis", *Proc. SPIE Conf. Vis. Comm., Boston, pages 987—998, 1993.*

[7] S. Qiu and H.G. Feichtinger, "The structure of the Gabor matrix and efficient numerical algorithms for discrete Gabor expansion." *Proc. SPIE VCIP94*, Chicago, pages 1146--1157, 1994.

[8] S.Qiu, H.G. Feichtinger, and T. Strohmer, "Inexpensive Gabor Decompositions." ***Proc. SPIE: Wavelets Applications in Signal and Image Processing*** pages 286-294, San Diego, 1994.

[9] T. Strohmer., "Computational frameworks for discrete Gabor analysis." ***Proc. SPIE: Advanced Signal Processing Algorithms, Architectures, and Implementations VI.*** San Diego, 1997.

[10] T. Strohmer, "A unified approach to numerical algorithms for discrete Gabor expansions." *Proc. SampTA - Sampling Theory and Applications,* Aveiro/Portugal, pages 297-302, 1997.

[11] S. Qiu, "Gabor-type matrix algebra and fast computations of dual and tight Gabor wavelets." *Opt. Eng.*, 36(1):276-282, 1997. 12.

[12] H.G. Feichtinger and T. Strohmer (Eds.). *Gabor analysis and algorithms. Theory and Applications,* Appl. Num. Harm. Anal., Birkhäuser Boston, 1998.

[13] S. Qiu., "Block-Circulant Gabor-matrix structure and discrete Gabor transforms." *Opt. Engrg.*, pp 2872--2878, 1995.

[14] S. Qiu and H.G. Feichtinger., "Discrete Gabor structure and optimal representation." *IEEE Trans. Signal Proc.*, 43(10):2258--2268, October 1995.

[15] L. S. Blackford, J.Choi, A. Cleary et.al.. *"ScaLAPACK Users' Guide"*, SIAM, Philadelphia, 1997.

[16] Manian Vidya, Vasquez Ramon "Efficient algorithms for discrete gabor transforms using multicomputer networks", Proceedings of SPIE, the International Society for Optical Engineering, 1996, vol. 2755, pp. 394-403