World Applied Sciences Journal 6 (1): 45-52, 2009
ISSN 1818-4952

Performance Analysis of Parallel Matrix Multiplication Algorithms Used in Image Processing

[1]Ziad Al-Qadi and [2]Musbah Aqel

[1]Al-Balqa Applied University, Amman, Jordan
[2]Applied Science University, Amman, Jordan

Abstract: In this paper, some basic image restoration problems are presented to introduce the importance of fast linear algebra algorithms. Then, an overview to some basic systolic arrays algorithms and mapping principles of these algorithms into systolic arrays is shown. Special attention has been done to the adaptive image filtering techniques. Moreover, the Singular Value Decomposition has been applied in a two-dimensional adaptive FIR filtering technique. However, a two-dimensional adaptive algorithm based on a Singular Value Decomposition (SVD) method will be presented using systolic arrays that is applied in the area of image processing.

Key words:

INTRODUCTION

In image processing we are dealing with deterministic and stochastic representations of images that improve the quality of images by removing degradation presented on image. In the process of the image restoration we try to restore an image from degraded one so that it is as close as possible to the original image. Some degradation contains random noise, interference, geometrical distortions, loss of contrast, blurring effects, etc...

Image restoration problem can be described as a problem of determining an appropriate inverse function to the degradation procedure. This is actually a two-side problem, first identifying the distortion function and then computing its inverse. Both can be combined into a single procedure. The most important problem is that image restoration is an ill-conditioned problem at best and a singular problem at worst.

For image restoration in a digital computer, it is assumed that the input images of the procedure are in discrete form. Several linear algebra tools may be applied to find the solution if the degradation is a linear procedure.

In all known methods, dealing with enormous data and fast and effective algorithms or structures have to be applied. The same problem arises in the area of image reconstruction, where a high resolution images should be reconstructed or object by processing data obtained from views of the object from many different perspectives where such a problem is a reconstruction of the 3-D object from 2-D projections in tomography. Convolution methods and Fourier transform techniques are extensively used in this area.

SYSTOLIC LINEAR ALGEBRA APPLICATIONS

An overview of the problem: Systolic array is defined as a connected set of processors with rhythmical data computation and propagation along the system [1]. In systolic arrays data is pumped from cell to cell among the array and the required computations are performed concurrently in the cells. All transforming procedures in systolic array can be grouped into the following classes [2]:

- Direct mapping from the algorithm-representation level to the systolic architecture,
- Mapping from the algorithm representation over algorithm model into hardware,
- Mapping of the previous designed architectures into a new architecture,
- Symbolic transformations and transformations.

Among the researchers S.Y.Kung's approach [4] is very popular, where the algorithm is presented by Signal Flow Graph (SFG). After some operations the resulting Signal Flow Graph with operation and delay modules maps straightforward into the systolic array.

Most modern DIP applications are based on linear algebra algorithms. In sequential algorithms the complexity of the algorithm depends on the required computation and storage capacity. The complexity analysis of the parallel algorithms includes another

Corresponding Author: Dr. Musbah Aqel, Applied Science University, Amman, Jordan

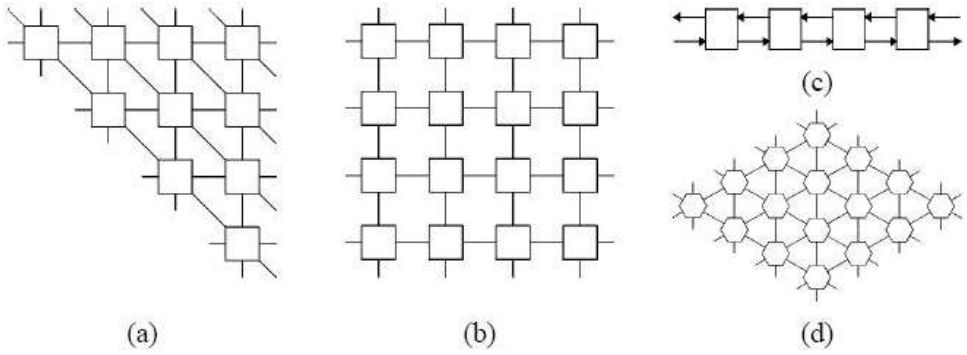

Fig. 1: Some examples of systolic arrays: (a) Triangular array, (b) square array, (c) BLA array, (d) hexagonal array

important parameter, the communication required. Therefore in massive parallel computation the most important factors are: computation, communication and memory.

Data distribution limitations and finite number of processing elements restrict our selves to a special class of applications, where recursions and the local dependency play very important role. These restrictions influenced the generality of the possible mapping procedures.

Systolic array algorithms: After tasks identification and possible VLSI architectures, new algorithms with degree of parallelism and regularity, with low communication overheads have to be developed [6].

Array algorithm is a set of rules solved with a finite number of steps on a multiple number of locally connected processors. The array algorithms are defined by synchronicity, concurrency control and granularity and communication geometry. A tool of systolic algorithms design has been proposed by Leiserson and Saxe [7].

This criterion defines a special class of algorithms that are recursive and locally dependent. The great majority of digital image processing algorithms possess such properties as shown in Fig. 1.

Basic linear algebra algorithms used for image processing: Digital image processing encompasses broad spectrum of mathematical methods. They are transform techniques, convolution, correlation techniques in filtering processes and set of linear algebraic methods like matrix multiplication, pseudo inverse calculation, linear system solver, different decomposition methods, geometric rotation and annihilation. Generally we can classify all image processing algorithms into two groups: basic matrix operations and special image processing algorithms. Fortunately, most of the algorithms fall in the classes of the matrix calculations, convolution, or transform type algorithms. These algorithms possess common properties such as regularity, locality and recursive ness.

In this paper, the speedup of a parallel algorithm is defined where it can be defined as a ratio of the corresponding sequential and parallel times. If we define:

- Np as number of processors
- Tn time required by the algorithm for n processors,
- T1 time required by the same algorithm for one processor, then the speedup is Ti/Tn>1

Another important parameter is efficiency of the calculation defined as T1/(NpTn)

Inner vector multiplication: Inner product of two n dimensional vectors x and y is close to this number of steps.

This product is obtained as product of the row vector u^T and the column vector v and can be given as:

$$y_i = \sum_{j=1}^{m} a_{ij} x_j$$

Sequentially it can be computed in (2n-1) steps, on parallel computer with n processors it can be computed in 1+log n steps. The speedup of the parallel version is approximately 2n/log(2n) and the achieved efficiency is 2/log(2n).

Matrix-vector multiplication: Matrix-vector multiplication algorithm of an n×m matrix A with a vector x of dimension m results in Y=Ax

Where y is an n element vector. The i-th element of y is defined as:

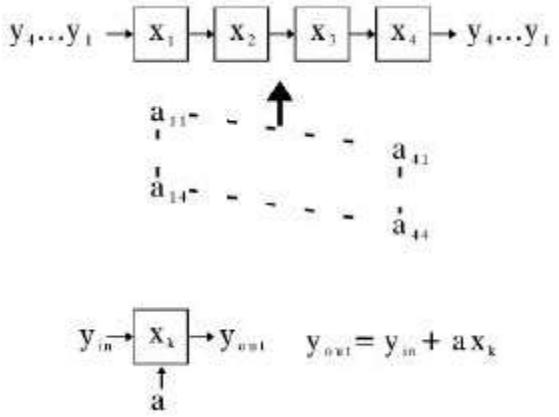

Fig. 2: Processor array for matrix-vector multiplication

$$yi = \sum_{j=1}^{m} a_{ij} x_j$$

where a_{ij} is of the matrix A. The Uniform Linear processor Array structure is convenient for this operation where one data stream is flowing to the right and the other data stream is flowing top down (Fig. 2).

The proposed parallel solution uses linear processor array with n processor elements required. The total execution time of the algorithm equals t=2n-1.

Matrix-matrix multiplication: Matrix-Matrix multiplication algorithm of an n×m matrix A with n×p matrix B results in new matrix denoted by C of dimension m×p. Matrix C is given by C=A·B where the elements are defined as:

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$$

This method can be realized with the array of processors of dimension m×p. The principle is the same as on Fig. 3. The connections are realized in horizontal and in vertical directions. Therefore the mesh connections of Linear processor Array structure is convenient for this operation where the data stream of matrix B is flowing to the right and the data stream of matrix A is flowing top down. The elements of matrix C are stored in the appropriate processors of the array. In the case of the matrix-matrix computation the expected speedup is

$$O\left(\frac{n^3}{\log n}\right)$$

Linear equation solvers: Solving a system of linear equations is one of the most important problems in DIP.

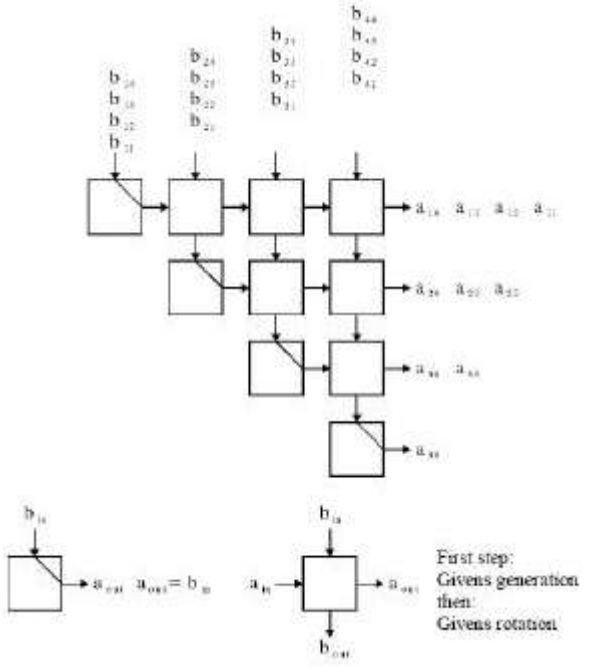

Fig. 3: Triangular array for QR decomposition

The problem is to find the solution vector x of dimension (n×1) for a given n linear equations Ax=y, where A is nonsingular matrix of dimension (n×n). The problem can be solved by computing an inverse matrix A^-1, that is A^-1 x=y. This inversion matrix computation procedure is computationally very intensive and procedure is numerically unstable. The approach using the triangularization procedure is often in use to triangularize matrix A. An upper triangular system $A^*x = y^*$, where is an n×n upper triangular system is finally solved by back-substitution.

In the numerical analysis literature there are many matrix triangularisation methods as Gauss elimination, QR and LU decomposition or other methods. Also other effective methods for solving the system of equations exist. They are bidiagonalization methods and Singular Value Decomposition methods.

Different techniques may be applied to obtain triangular matrix decomposition. The most commonly used are methods using Givens rotations or Householder reflections. Although Householder reflections are proven to be more efficient in sequential algorithms, this is not the case for parallel execution. Using O(n) processors, direct implementation of Householder's reduction and the Gram-Schmidt algorithm require O(n.log n) steps. Given's reduction can be modified to produce a parallel algorithm in O(n) steps with the same number of processors. The QR tridiagonization procedure uses Givens rotations to annihilate lower triangular elements. For each

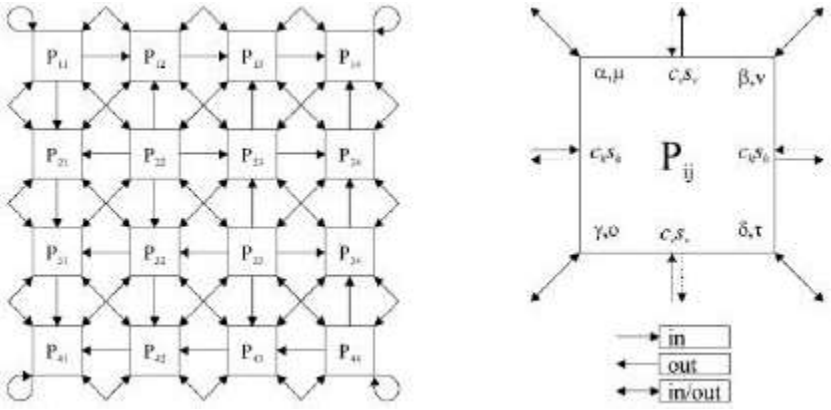

Fig. 4: Systolic array implementation of the Jacobi decomposition

annihilation, one rotation is to be performed. The entire process of tridiagonization could be written as:

$$R = Q^T A$$

$$Q^T = Q_1 \cdot Q_2 \cdot \ldots \cdot Q_k \cdot \ldots \cdot Q_n$$

$$Q_k = Q^{(k,k+n)} \cdot \ldots \cdot Q^{(k,n)}$$

$$Q^{(k,j)} = \begin{bmatrix} 1 & & & 0 \\ & \cos\theta_{ki} & \sin\theta_{ki} & \\ & -\sin\theta_{ki} & \cos\theta_{ki} & \\ 0 & & & 1 \end{bmatrix}$$

$$\theta_{ki} = \arctan\frac{a_{ki}}{a_{kk}}$$

After the algorithm has been transformed into a system of uniform recurrence equations, the mapping to a systolic structure is straightforward. The result is a triangular systolic array, as shown on Fig. 3.

Two different purpose processor elements are used. Elements on the diagonal are simply delay elements used to transfer the values of b coming from the top to the right. Other elements perform Givens parameter generation in the first operational step and Givens rotations afterwards. The results can be obtained from the right side of the array.

Actually, n(n-1)/2 processor elements are required, as the delay elements on the diagonal of the array are can simply be realized using registers instead of processor elements.

Another important methods in image processing are eigenvalue/eigenvector and singular value/vector decomposition methods. Some parallel algorithms have been developed like parallel version of the Jacobi and Jacobi-like algorithms, QR algorithm for obtaining several eigenvalues of a symmetric tridiagonal matrix [10], etc.

Jacobi algorithm is described in Golub [11] and in Wilkinson, Reinsch [12]. A real symmetric matrix A can be reduced to the diagonal form by a sequence of plane rotations. In practice this iterative process of reduction of the off-diagonal elements is terminated when these of-diagonal elements become negligible comparing to the elements on the main diagonal. Classical Jacobi algorithm eliminates the element in the position (p, q) and its symmetric counterpart. The main task is to find a sequence of reduction the off-diagonal element in parallel, where we are not concerned about destroying zeros that we previously introduced. It is possible to eliminate more than one element simultaneously in one sweep. Maximal number of the annihilated off-diagonal elements in one sweep is $(n^2-n)/2$ pairs. In approximately few (8) sweeps the matrix becomes practically diagonal. The diagonal elements represent the eigenvalues ant he products of individual transformations are taken as the eigenvectors. In the structure of $O(n^2)$ processors, one sweep requires O(n) steps yielding a speedup over sequential algorithm of $O(n^2)$. The suggested array is shown on Fig. 4.

Other methods reduce the matrix to a tridiagonal form or upper Hessenberg form, depending if matrix is symmetric or not. If the matrix is symmetric tridiagonal, we may apply the QR algorithm. This method is described in Reisch Wilkinson [12].

Singular Value Decomposition of matrices is useful in multidimensional image processing. Matrix A can be factorized in $Q_1 \Sigma Q_2^T$ Q, where Q1 $_1$ is an mxm orthogonal matrix and Q_2 is an nxn orthogonal matrix and S has the diagonal form

$$\Sigma = \begin{vmatrix} D & 0 \\ 0 & 0 \end{vmatrix} \text{ where } D = \text{diag}(\sigma_1, \sigma_2, \ldots, \sigma_n)$$

$$\sigma_1 \geq \sigma_2 \geq \ldots \sigma_r \geq 0 \text{ and r is rank of matrix A}$$

$$\text{Theform} A = Q_1 \Sigma Q_2^T = \sum_{i=1}^{r} \sigma_i u_i v_i^T$$

is called the SVD of the matrix A, where the singular values σ_i are the square roots of the none zeros eigenvalues of A^T and u_i and v_i are column vectors of the matrices Q_1 and Q_2 respectively. The column vectors of Q_1 and Q_2 are the eigenvectors of A^TA.

The preferable method for solving the SVD problem is described in Golub and Van Loan [11]. The described technique finds U and V simultaneously by simply applying the symmetric QR algorithm to A^TA. This method can be also applied for solving the common problem in signal and image processing, the least square problem.

THE SVD AND SYMMETRIC EIGENVALUE PROBLEMS

Singular value decomposition (SVD) of a real m by n matrix A is its factorization into the product of three matrices:

$$A = U\Sigma V^T \qquad (3.1)$$

where U is an m by n matrix with orthonormal columns, Σ is an n by n nonnegative diagonal matrix and V is an n by n orthogonal matrix (we assume here that m >= n).

The diagonal elements σ_i of Σ are the singular values of A. The singular value decomposition is extremely useful in digital image processing. The SVD is usually computed by a two-sided orthogonalization process, e.g. by two-sided reduction to bidiagonal form (possibly preceded by a one-sided QR reduction) followed by the QR algorithm. On a systolic array it is simpler to avoid bidiagonalization and to use the two-sided orthogonalization method of Kogbetliantz *et al*. [14-16] rather than the standard Golub Kahan Reinsch algorithm. However, it is even simpler to use a one-sided orthogonalization method due to Hestenes. The idea of Hestenes is to iteratively generate an orthogonal matrix V such that AV has orthogonal columns. Normalizing the Euclidean length of each nonnull column to unity, we get

$$AV = \tilde{U}\Sigma \qquad (3.2)$$

As a null column of $\tilde{U}$ is always associated with a zero diagonal element of Σ, there is no essential difference between (3.1) and (3.2).

There is clearly a close connection between the Hestenes method for finding the SVD of A and the classical Jacobi method for finding the eigenvalues and eigenvectors of ATA. This is discussed in Section 3.4.

Implementation of the Hestenes method: Let A1 = A and V1 = I. The Hestenes method uses a sequence of plane rotations
Qk chosen to orthogonalize two columns in Ak+1 = AkQk. If the matrix V is required, the plane rotations are accumulated using Vk+1 = VkQk. Under certain conditions (Discussed below) limQk = I, lim Vk = V and limAk = AV. The matrix Ak+1 differs from Ak only in two columns, say columns i and j. In fact

$$\left(a_i^{(k+1)}, a_j^{(k+1)}\right) = \left(a_i^k, a_j^k\right)\begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}$$

where the rotation angle θ is chosen so that the two new columns $a_i^{(k+1)}$ and $a_j^{(k+1)}$ are orthogonal. This can always be done with an angle θ satisfying

$$|\theta| \le \pi/4 \qquad (3.3)$$

It is desirable for a "sweep" of n(n-1)/2 rotations to include all pairs (i, j) with i < j. On a serial machine a simple strategy is to choose the "cyclic by rows" ordering (1, 2), (1, 3), · · ·, (1, n), (2, 3), · · ·, (n - 1, n).

It has been shown [17] that the cyclic by rows ordering and condition (3.3) ensure convergence of the Jacobi method applied to A^TA and convergence of the cyclic by rows Hestenes method follows.

The symmetric eigenvalue problem: As noted above, there is a close connection between the Hestenes method for finding the SVD of a matrix A and the Jacobi method for finding the eigenvalues of a symmetric matrix B = A^TA An important difference is that the formulas defining the rotation angle θ involve elements bi,j of B rather than inner products of columns of A and transformations must be performed on the left and right instead of just on the right (since $(AV)^T$ (AV) = V^T BV) instead of permuting columns of A, we have to apply the same permutation to both rows and columns of B. This is easy if we use a square systolic array of n/2 by n/2 processors with nearest-neighbor connections (assuming, for simplicity, that n is even). If less than n^2/4 processors are available, we can use the virtual processor concept. For example, on a linear systolic array with P <= n/2 processors, each processor can simulate ~ n/(2P) columns of n/2 virtual processors.

Similarly, on a square array of P _ n2/4 processors, each processor can simulate a block of ~ n2/(4P) virtual

processors. In both cases, communication paths between virtual processors map onto paths between real processors or communication internal to a real processor.

Other SVD and eigenvalue algorithms: As shown above, the Hestenes method could be used to compute the SVD of an m by n matrix in time O(mn^2S/P) using P = O(n) processors in parallel.

Here S is the number of sweeps required (conjectured to be O(log n)). In Section 3.4 we sketched how Jacobi's method could be used to compute the eigen-decomposition of a symmetric n by n matrix in time O(n^3S/P) using P = O(n^2) processors. It is natural to ask if we can use more than (n) processors efficiently when computing the SVD. The answer is yes and this can be used to compute the SVD of a square matrix using a parallel algorithm very similar to the parallel implementation of Jacobi's method. The result is an algorithm which requires time O(n^3S/P) using P = O(n^2) processors.

In order to find the SVD of a rectangular m by n matrix A using O(n^2) processors, we first compute the QR factorization QA = R and then compute the SVD of the principal n by n sub matrix of R (i.e. discard the m - n zero rows of R). It is possible to gain a factor of two in efficiency by preserving the upper triangular structure of R.

The Hestenes/Jacobi/Kogbetliantz methods are not often used on a serial computer, because they are slower than methods based on reduction to bidiagonal or tridiagonal form followed by the QR algorithm. Whether the fast serial algorithms can be implemented efficiently on a parallel machine depends to some extent on the parallel architecture. For example, on a square array of n by n processors it is possible to reduce a symmetric n by n matrix to tridiagonal form in time O(n log n). On a serial machine this reduction takes time O(n^3). Thus, a factor O(log n) is lost in efficiency, which roughly equates to the factor O(S) by which Jacobi's method is slower than the QR algorithm on a serial machine. It is an open question whether the loss in efficiency by a factor O(log n) can be avoided on a parallel machine with P = (n^2) processors. When P = O(n), "block" versions of the usual serial algorithms are attractive on certain architectures and may be combined with the "divide and conquer" strategy.

LS SVD DIGITAL IMAGE FILTERING

For illustration, the use of singular value decomposition in two-dimensional filtering applications will be presented. First, the Wiener solution will be extended to two-dimensional problem, introducing special formulation of an image signal matrix. The problem will be solved algebraically with two-dimensional convolution filter implemented. The Wiener normal equation will be solved by using singular value decomposition of the image signal matrix. The effectiveness of the suggested method will be illustrated on a practical filtering problem.

Image restoration: We have decided to represent the degradation model for our imaging system in a form of discrete linear point-spread degradation functions. For discrete image F degraded to image G and subjected to additive noise N, we may write

$$g(x,y) = \sum_{u=1}^{N}\sum_{v=1}^{N} h(x,y,u,v)f(u,v) + n(x,y)$$

or, alternatively, in tensor notation

$$[G] = [[H]]\{[F]\}+[N]$$

with two-dimensional matrices G, F and N and using the four-index operator H [22].

The objective of restoration is to find an inverse to the degradation function. The solution presented is not valid for all cases of image degradation. In some cases it is possible to use convolution filter to restore the image. The solution may then be represented in a form of $\hat{F} = W**G$ with symbol ** standing for 2-D convolution. We have to point out that the solution in such a formulation exists only for linear, space-invariant distortion functions with finite (space-limited) response. The general adaptive filter representation for this case is illustrated in Fig. 4.

The filter operates on a real image (matrix) X that is corrupted with noise. The desired signal (reference image) is also provided. The filtering parameters can be represented in form of an N×N matrix W and the filtering process may be represented by convoluting the image input X with the matrix W. During the adaptation, the filtering weights may be changed in order to obtain optimal solution. The filtering result is given by

$$\hat{f}(x,y) = \sum_{i=0}^{M}\sum_{j=0}^{M} w(i,j)g(x+1-k, y+1-k)$$
$$M = 2k+1$$

The difference between the desired and the resulting image

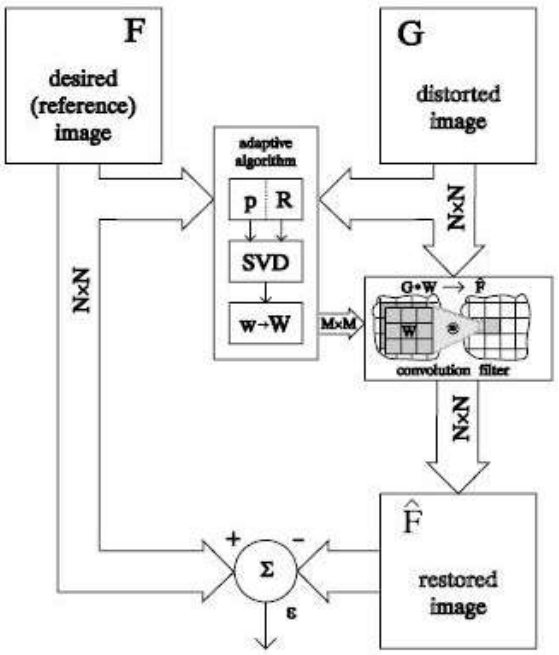

Fig. 4: SVD based 2-D adaptive filter

$$e(x,y) = f(x,y) - \hat{f}(x,y)$$

is called the estimation error. From Wiener filter theory, optimal filtering coefficients W are defined by the minimum mean-square error criteria. The objective function $J(W) = E[e^2(x, y)]$ should be minimized for W to obtain the optimum filter.

For this particular example, Wiener optimal solution is to be applied. The idea is well known from 1-D adaptive filtering, where instantaneous estimates of gradient of the error surface J(W) are used to approach the optimum solution iteratively. The algorithm is popularly called LMS algorithm. It is possible to extend the algorithm to be used in both x and y image dimensions, iteratively searching for the solution either column wise or row wise. The procedure is numerically convenient due to low storage and computing requirements. The problem of this approach is that the instantaneous estimates of the error surface have relatively large variances. The estimate of their gradient vectors may then not always be pointing to a global optimum; the fact could cause unstable performance of the algorithm. The stability may be improved using smaller adaptation step-size, however this seriously affects the convergence rate of the procedure.

As already suggested, the normal equation $Rw_0 = p$ is to be solved for w, using special inversion techniques. The one proven to be very efficient is the method using SVD matrix decomposition.

Singular value decomposition: One of the methods for the stable inversion process of the matrix $R = X^TX$ is called SVD pseudo-inversion, which is better than to calculate straight inverse of $R^{-1} = X^TX^{-1}$. The solution for W can be expressed directly as $w_0 = X^+d$ where pseudo-inverse X^+ is defined in terms of the products of the singular-value decomposition of $U^TXV = \Sigma$ of X. The procedure is numerically stable and its solution is unique in that its vector norm is minimum [18].

Convolution operator W can be created by restacking the values of the vector w back to the MxM matrix form:

$$w = \sum_{\substack{i=1 \\ \sigma_i \neq 0}}^{M^2} \frac{u_i^T d}{\sigma_i} v_i \quad W = \begin{bmatrix} w(1) & \cdots & w(M) \\ \vdots & \ddots & \vdots \\ w(M(M-1)+1) & \cdots & w(M^2) \end{bmatrix}$$

The non-iteratively calculated filtering parameters are optimal for the specific image/distorted image combination. They may be directly applied in a classical two-dimensional convolution filter.

Implementation of the procedure: The procedure may be implemented as a systolic array algorithm. The actual algorithm is to be combined out of partial linear algebra solutions presented above. Note that the array to perform singular value decomposition is almost identical to eigendecomposition array.

The simulation results show that the Wiener filtering principle can successfully be implemented in image restoration. Methods well known from the linear algebra theory that may be applied instead of classical methods based on Fourier transformation. The effectiveness of the procedure may be improved using special updating techniques.

CONCLUSIONS

Characteristically for almost all presented linear algebra operations that suggested already have been used in digital image processing applications are consisting of a huge number of relatively basic mathematical operations. The fact that the operations are repetitive, yet applied on a wide set of data inspired us to employ several processor elements performing the same task on separate data elements in parallel. Special properties of the mentioned processing problem allow us to construct a massive array of equal processor elements, which concurrently perform the necessary numerical operations.

There exist several well-known parallel computer architectures; the architecture may vary according to the applied processor elements, reconfigurability, data

interchange connections, etc. The architecture to be applied on a specific problem depends mostly on a problem itself. As the digital image processing demands high speed computing with fixed procedures in use at relatively low cost, general-purpose parallel computers are not convenient for use. Digital image processing is a data-oriented computing problem, so architectures with global data interchange are to be omitted. What we really need is an array of locally interconnected processor elements with local memory. The processor elements should synchronously perform the same set of operations on the data structure. This architecture is a systolic array - the rhythmical operation of the array reminds us to the systolic of the heart

The basic approach to mapping techniques and some possible applications were presented in this chapter. However, this was only a brief introduction to the world of special-purpose VLSI systolic architecture. More details on the described procedures as well as on optimization techniques not presented here may be found from the literature.

REFERENCES

1. Kung, H.T. and C.E. Leiserson, 1978. Systolic Arrays (for VLSI), Tech. rep. CS-79-103, Carneige Mellon University, Pitsburg, PA.
2. Fortes, A.B., K.S. Fu and B.W. 1985. Wah, Systematic approaches to the design of algorithmic specified systolic arrays. Proc. IEEE ICASSP´85, IEEE Computer Society Press, pp: 300-303.
3. Cappello, R. and K. Stieglitz, 1983. Unifying VLSI array designs with geometric transformations. Proc. of 1983 Int. Conf. on Parallel Processing, pp: 448-457.
4. Kung, S.Y., 1988. VLSI Array Processors, Prentice Hall.
5. Kung, S.Y., K.S. Arun, R.J. Gal-Ezer and D.B. Rao, 1982. Wavefront array processor: Language, architecture and applications. IEEE Tr. Comput., c-31 1982, J. Tasic, U. Burnik, pp: 1054-1066.
6. Gušev, M., 1992. Processor Array Implementations of Systems of Affine Recurrence Equations for Digital Signal Processing. Ph.D dissertation, University of Ljubljana.
7. Leiserson, C.E. and J.B. Saxe, 1983. Optimizing synchronous systems, J.VLSI and Computer Systems, pp: 41-67.
8. Kung, S.Y., 1980. VLSI array processor for signal processing. Conf. Advanced Res. in Integrated Circuits, MIT, Cambridge.
9. Kung, H.T., 1983, Notes on VLSI computation. Parallel Processing Systems, Evans, D.J. (Ed.). Cambridge University Press, pp: 339-356.
10. Sameh, 1977. Numerical Parallel Algorithms - A Survey; High Speed Computer and Algorithm Organization, Academic Press, pp: 207-228.
11. Golub, H. and C.F. Van Loan, 1989. Matrix computations, John Hopkins University Press, Baltimore, London.
12. Wilkinson, H., 1965. The Algebraic Eigenvalue Problem, Oxford University Press, London.
13. Moldovan, I., 1982. On the analysis and synthesis of VLSI algorithms. IEEE Trans. Comput., 31: 1121-1126.
14. Brent, R.P., F.T. Luk and C. Van Loan, 1985. Computation of the Singular Value Decomposition using mesh-connected processors. J. VLSI and Computer Systems, 1 (3): 250-260.
15. Brent, R.P. and F.T. Luk, 1985. The solution of singular-value and symmetric eigenvalue problems on multiprocessor arrays. SIAM J. Sci. Stat. Comput., Vol: 6 (1).
16. Thiele, L., 1988. Computational arrays for cyclic-by rows Jacobi Algorithms. SVD and Signal Processing. Algorithms, Applications and Architectures, Elsevier Science Publishers B. V. North Holland.
17. Burnik, U., G. Cain and J. Tasic, 1993. On the Parallel Jacobi Method Based Eigenfilters. COST 229 WG4 Workshop on Parallel Computing, Funchal, Portugal.
18. Haykin, S., 1989. Modern Filters, Macmillan, New York.
19. Haykin, S., Adaptive Filter Theory. Prentice Hall, Englewood Cliffs, N.J.
20. Tasic, J. *et al*., 1993. Eigenanalysis in Adaptive FIR Filtering. Internal Report, University of Westminster.
21. Gonzales, R.C. and R.C. Woods, 1992. Digital Image Processing, Addison-Wesley Publishing Company.
22. Andrews, H.C. and B.R. Hunt, 1977. Digital Image Restoration, Prentice Hall, Englewood Cliffs, New Jersey.