Linear Array Implementation of the EM Algorithm for PET Image Reconstruction

K. Rajan, L. M. Patnaik, *Fellow, IEEE*, and J. Ramakrishna

Abstract—The PET image reconstruction based on the EM algorithm has several attractive advantages over the conventional convolution backprojection algorithms. However, the PET image reconstruction based on the EM algorithm is computationally burdensome for today's single processor systems. In addition, a large memory is required for the storage of the image, projection data, and the probability matrix. Since the computations are easily divided into tasks executable in parallel, multiprocessor configurations are the ideal choice for fast execution of the EM algorithms. In this study, we attempt to overcome these two problems by parallelizing the EM algorithm on a multiprocessor system. The parallel EM algorithm on a linear array topology using the commercially available fast floating point digital signal processor (DSP) chips as the processing elements (PE's) has been implemented. The performance of the EM algorithm on a 386/387 machine, IBM 6000 RISC workstation, and on the linear array system is discussed and compared. The results show that the computational speed performance of a linear array using 8 DSP chips as PE's executing the EM image reconstruction algorithm is about 15.5 times better than that of the IBM 6000 RISC workstation. The novelty of the scheme is its simplicity. The linear array topology is expandable with a larger number of PE's. The architecture is not dependent on the DSP chip chosen, and the substitution of the latest DSP chip is straightforward and could yield better speed performance.

I. INTRODUCTION

POSITRON Emission Tomography (PET) is an imaging technique to visualize the spatial and temporal distribution of the radio-nuclides inside the human body by measuring the event counts of positron-electron annihilation. There are two main approaches for PET image reconstruction: analytic methods such as the Convolution Back Projection (CBP) algorithm [13] which was originally devised for computer aided tomography (CAT), and the iterative algorithms such as expectation maximization (EM) algorithms. An analytic algorithm usually consists of two main computations. One is filtering and the other is back projection. An iterative algorithm, on the other hand, starts with an initial guess of the solution and iteratively updates (corrects) the object according to the computed pseudo-projection and the measured projection data, till convergence is reached. The stopping rule proposed in [18] is based on a statistical approach, where after each iteration, the estimate is accepted or rejected as

the final image based on the result of a statistical hypothesis test. The major computations in an iterative method are forward (pseudo-projection) and backward (correction) projections. The EM algorithm requires longer computation time than the CBP method. However, the image reconstructed using the EM algorithm is less noisy than the CBP image and the EM algorithm does not require the projection data to be equally spaced.

Various efforts have been made to speed up the image reconstruction tasks. These efforts essentially fall into three categories, i.e., algorithmic improvement, dedicated hardware, and parallel processing. In the first category, most of the attempts for iterative reconstruction methods have been concerned with reduction of the number of iterations, i.e., to make convergence faster [7], [10], [13], [15]. In the second category, dedicated hardware techniques have been employed to speed up the computation [8], [9], [17]. To overcome the two major problems that impede the routine use of the EM algorithm for clinical use, i.e., the long computation time and very large memory requirement, it is imperative to rely on parallel processing techniques which have a potential to speed up the reconstruction by several orders of magnitude. Several attempts at improving the speedup using multiprocessor approach have been reported [2], [3], [5], [6]. Chen *et al.* have studied the parallelization of the EM algorithm on a message passing system (Intel iPSC/2) and on a shared memory system (BBN Butterfly GP 1000) [2]. A data and task partitioning scheme called partition-by-box is proposed in this study. The *partition-by-box* scheme proposed by Chen uses the broadcast and partial result integration algorithms. The binary tree architecture is more efficient to perform the broadcast and integration algorithms. Though Chen *et al.* have used the iPSC/2 hypercube system, the pseudo-binary tree embedded in the hypercube has been used for the EM algorithm. In [3], Chen *et al.* have proposed new integration and broadcasting algorithms for hypercube, ring, and n-D mesh topologies, which are more efficient than conventional algorithms. A close look at the EM reconstruction algorithm shows that most of the computation time is spent in executing *multiply* and *multiply-accumulate* types of instructions. Digital Signal Processors (DSP's) are optimized processors to execute fast *multiply* and *multiply-accumulate* instructions. In our earlier study [14], we investigated the implementation of the parallel EM algorithm on an Extended Hypercube (EH) topology. The EH is a hierarchical, expansive, recursive structure with a constant predefined building block. The EH (k, l) (l is the degree of the EH) is built using basic modules consisting of a k -cube of processor elements (PE's) and a Network Controller

Manuscript received July 13, 1994; revised December 6, 1994 and April 6, 1995.

K. Rajan is with the Department of Physics, Indian Institute of Science, Bangalore 560 012, India.

L. M. Patnaik is with Microprocessor Applications Laboratory, Indian Institute of Science, Bangalore 560 012, India.

J. Ramakrishna is with the Department of Physics, Indian Institute of Science, Bangalore 560 012, India.

IEEE Log Number 9413066.

(NC). The NC is used as a communication processor to handle intermodule communication; 2^k such basic modules can be interconnected via 2^k NC's, forming a k -cube among the NC's. In general, an EH (k, l) consists of one NC at the l th level, and a k -cube of 2^k NC's/PE's at the $(l-1)$ st level. The NC's/PE's at the $(l-2)$ st level of hierarchy form 2^k distinct k -cubes. Thus, we have k -cubes at all levels j for $0 \leq j \leq l$. An EH (3, 1) with ADSP 21020 DSP devices as PE's was implemented. Eight DSP chips formed the 3D cube and one DSP chip was configured as an NC; the NC was connected to all the eight PE's through direct links. The PE/NC link was created through a message channel generated out of a dual-port RAM (DPR). This DPR memory channel allows overlapped computation and communication tasks. In addition, the EH executes the integration of partial results efficiently. The EH topology supports efficient single node broadcast and multi-node broadcast. These features were used in the execution of the EM algorithm. It was found that the computational speed performance of an EH (3, 1) using DSP chips as PE's executing the EM image reconstruction algorithm is about 100 times better than that of the IBM RISC 6000 workstation for image sizes of 16×16 , 32×32 , 64×64 , and 128×128 . However, the algorithmic complexity is much higher in EH. Also hardware complexity of the EH with a memory channel is very high. Here in this study, the implementation of the parallel version of the EM algorithm on a linear array of processors was investigated. The novelty of the scheme is its simplicity. The linear array topology is very easy to build. The expansion of the linear array is quite straightforward. With the announcement of latest DSP devices with serial and link ports, it is possible to have glueless interconnection of PE's to generate large networks. The algorithm also breaks up into simple subtasks.

II. THE EM ALGORITHM

The EM algorithm is the basic approach used to maximize the log likelihood objective function for the PET image reconstruction problem. PET images are used to study the human physiology and organ functions. The patient is given a tagged substance which emits positrons. Each positron annihilates with an electron and emits two photons in opposite directions. The patient is surrounded by a ring of detectors and the two photons are detected in time coincidence by a pair of detector elements defining a detector unit or detector tube. The reconstruction problem in PET is to determine the memory map of the annihilations from which information about the regional physiology can be obtained.

In the recent literature, much attention is given to maximum likelihood reconstruction based on expectation maximization (EM). These algorithms are appealing because, unlike other methods such as CBP, they take into account the statistical nature of the measurements. Dempster *et al.* [4] presented a general algorithm to produce maximum likelihood estimates from incomplete data. Shepp and Vardi [16], and Lange [11] applied this technique to image reconstruction from PET measurements. The measurement system is shown in Fig. 1.

Fig. 1. The PET measurement system.

The EM algorithm for image reconstruction can be written as [16], [11]

$$\lambda[i]^{(u+1)} = \frac{\lambda[i]^{(u)}}{\sum_{j=1}^{N_t} p(i, j)} \sum_{j=1}^{N_t} \frac{y[j] p(i, j)}{\sum_{i'=1}^N \lambda[i']^{(u)} p(i', j)}, \quad i = 1, \dots, N \quad (1)$$

where

- $\lambda[i]$ number of photon pairs emitted from box i (the image to be reconstructed),
- u iteration index,
- $p(i, j)$ probability that a photon emitted from box i is detected by tube j ,
- $y[j]$ number of photon pairs detected by tube j (projection data),
- N total number of reconstruction boxes, and
- N_t total number of detector tubes.

The standard EM iteration step given by (1) can be rewritten in an additive form [12] as

$$\lambda[i]^{(u+1)} = \lambda[i]^{(u)} + \frac{\lambda[i]^{(u)}}{\sum_{j=1}^{N_t} p(i, j)} \sum_{j=1}^{N_t} \frac{y[j] - \sum_{i'=1}^N \lambda[i']^{(u)} p(i', j)}{\sum_{i'=1}^N \lambda[i']^{(u)} p(i', j)} p(i, j), \quad i = 1, \dots, N. \quad (2)$$

Equation (2) has been implemented on the linear array system. The EM algorithm converges toward a possible unique minimum, and the image obtained after convergence is independent of the initial estimate $\lambda[i]^0$. However, if the procedure is stopped before the maximum likelihood is reached, the initial estimate can strongly influence the result. All the reconstructions carried out in this study were started with identical $\lambda[i]$ values.

III. COMPUTATIONAL COMPLEXITY

The complexity of the EM algorithm is given in [16]. For a 128×128 square object, there are 16384 object boxes. The probability $p(i, j)$ that an emission in box i is detected in a tube j depends on a number of physical factors such as the geometry of the measurement system, the decomposition of the object space, the physical properties of the medium and the response of the detector system. In this study, it is assumed that the probability of an emission in box i and its detection in tube j depend only on the geometry of the measurement system. In such a case an annihilation event in box i is detected in a tube j with the probability $p(i, j)$ proportional to the angle of view from the center of the box i in to the detector tube j . Shepp *et al.* [16] have shown that the choice of $p(i, j)$ based only on the geometry of the measurement system is reasonable, and that the results of the reconstruction do not depend critically on the choice of $p(i, j)$. Since there are N_t number of detector tubes and N object boxes, the dimension of probability matrix $p(i, j)$ is $N \times N_t$. For a circular ring measurement geometry with N_d detector elements equally spaced around the circle of radius $\sqrt{2}$ circumscribing the display boxes, the total number of detector tubes N_t is given by $N_t = (N_d/2) * (N_d/2 + 1)$ since there are $(N_d/2 + 1)$ detector intervals opposite each one [16]. For a system with 128 detectors and an object space of 128×128 , the dimension of the probability matrix $p(i, j)$ is 4160×16384 . Since only a very small percentage of the total detector tubes passes through a box, the probability matrix $p(i, j)$ is highly sparse. To minimize the storage requirement, only the nonzero $p(i, j)$'s are stored.

The EM algorithm represents a family of important problems that are rich in data parallelism. The data parallelism in the EM algorithm may be described by two spaces namely, *box* and *tube* spaces. There are three possible data and task partitioning schemes [2] to solve the image reconstruction problem based on the EM algorithm. The first approach, the *partition-by-box scheme*, is based on partitioning the EM algorithm based on the box space. In this approach, for both the forward and backward steps, a box and all the task and data associated with that box are assigned to a PE.

The second approach, *partition-by-tube scheme*, uses the detector tube as a subpartition. In this scheme, for both forward and backward steps, a detector tube is assigned to a PE. So all the task and data associated with that tube are assigned to a PE. The partition-by-box and partition-by-tube schemes give almost similar performance. The potential problem of the partition-by-box scheme and partition-by-tube is that the computational load is not well balanced. For the partition-by-box scheme, the computational load associated with each box in the partition region is different. For the partition-by-tube scheme, the number of pixels in each of the detector tubes is different.

In this study, we use the partition-by-tube scheme for implementation on a linear array. The same logic can be used for the implementation of the partition-by-box scheme on a linear array.

The third approach, the *partition-by-tube-and-box scheme* uses the partition-by-tube scheme for the forward step, and

the partition-by-box scheme for the backward step. So the storage overhead for the partition-by-tube-and-box scheme is roughly twice that of the other two schemes, because we have to store the task and data corresponding to the tube space for the forward step, and the box space data and task for the backward step.

In order to implement the partition-by-tube scheme, two approaches are possible. First, the indices of the pixels in each of the N_t tubes are precomputed and stored. Second, the indices of the pixels in each of the tubes are computed in each step. The first approach is faster, but it requires more memory. Most of the memory is required to store the indices of the pixels in each of the N_t tubes, the corresponding pixel values, and the pixel_tube_probabilities. The parallel algorithm formulated is based on the first approach. We introduce three 1D arrays: integer pixel_indices[] to hold the indices of the pixels in each of the N_t tubes, the corresponding real pixel_tube_probability[], and integer pixel_count[] to hold the number of pixels in each tube. The array pixel_indices[] is stored in numerically increasing order to use binary search to check if a particular pixel lies in the tube processed by that PE.

The partition-by-tube scheme can be easily divided into tasks executable in parallel. In this study, we are investigating the implementation of the partition-by-tube scheme on the linear array multiprocessor system. Each PE_{*t*} in the linear array executes the following algorithm:

linear array implementation of the EM algorithm based on partition-by-tube

for each detector tube $t = 1$ to N_t

```

begin
  for  $i = 1$  to  $N$  /*  $\Delta P$  computation */
    begin
      receive  $\lambda[i]$  from PE $t-1$ 
      transmit  $\lambda[i]$  to PE $t+1$ 
      update the index of the pixel in PE $t$  register
      binary search 1D array pixel_indices[]
      if pixel index matches with any element
        in the array pixel_indices[]
          get the corresponding pixel_tube_probability  $p$ 
          pseudo_projection[t]
          := pseudo_projection[t] +  $\lambda[i] \cdot p$ 
    end
     $\Delta P[t] = \frac{(\text{projection}[t] - \text{pseudo\_projection}[t])}{\text{pseudo\_projection}[t]}$ 
  for  $i = 1$  to  $N$  /* correction phase */
    begin
      receive  $\lambda[i]$  from PE $t-1$ 
      update the index of the pixel in PE $t$  register
      binary search 1D array pixel_indices[]
      if pixel index matches with any element
        in the array pixel_indices[]
          get the corresponding pixel_tube_probability  $p$ 
           $\lambda[i] = \lambda[i] + \lambda[i] \cdot \Delta P[t] \cdot p$  /* partial pixel
          value update */
          transmit  $\lambda[i]$  to PE $t+1$ 
    end
  end.
```

Fig. 2. Linear array multiprocessor system.

The image reconstruction algorithm based on the EM technique is iterative in nature. The major computations in an iterative scheme are forward (pseudo-projection) and backward (correction) operations. From an estimate of the object, the forward algorithm finds the pseudo-projection on each detector tube j . The ratio of the difference between the projection data (actual measurement data) and the pseudo-projection data (computed value) to the pseudo-projection data, i.e., $\Delta P(t)$, gives the extent of resemblance between projection of the object and the projection of the reconstructed object. Based on ΔP , corrections are made to the initial guess of the object. In the correction operation, ΔP in the projection domain is back-projected onto the object domain.

IV. MULTIPROCESSOR ARCHITECTURE

The multiprocessor architecture used for the implementation of the parallel EM algorithm is shown in Fig. 2. It consists of linear connection of N_t identical processors. PE_j is connected to PE_{j-1} and PE_{j+1} through a set of high speed first-in-first-out (FIFO) buffers. PE_j can send a 32-b data item to PE_{j+1} and receive a 32-b data item from PE_{j-1} through the FIFO's. In computing the forward step (pseudo-projection computation), once the pipeline is filled, all the N_t identical processors operate in parallel with each processor computing the pseudo-projection on a tube assigned to that PE. There are N_t number of detector tubes; the task and data corresponding to tube j are preloaded onto PE_j . Each PE holds four 1D arrays: integer `pixel_indices[]` that hold the indices of the boxes that lie in tube j , real `pixel_values[]` of the pixel indices stored in the array `pixel_indices[]`, the corresponding real `box_tube_probability[]`, and integer `pixel_count[]` that holds the number of boxes in tube j . It also gets preloaded with the projection data corresponding to the tube it processes. For maximum processing speed, there should be one processor per detector tube; otherwise each processor will hold data corresponding to multiple detector tubes. In a configuration with N_p processors and N_t detector tubes, each processor will be assigned with data and task corresponding to N_t/N_p tubes. In order to minimize the potential load imbalance problem, data and task corresponding to N_t tubes in N_p PE's were distributed cyclically.

The host processor sends the pixel intensities to the processor array. PE_j receives the box intensity, and sends the intensity data to the neighboring node, PE_{j+1} . Each PE_j keeps track of the index of the pixel it currently processes using a processor register. PE_j computes the contribution to the

pseudo-projection on tube j from the current box intensity. The integer array `pixel_indices[]` are stored in numerically increasing order. So each PE_j uses binary search to find out whether the current pixel lies in tube j . If the current pixel lies in tube j , the corresponding probability value is used to compute the contribution to the pseudo-projection on tube j . Once PE_j completes the processing of a pixel, it receives the next pixel intensity data from PE_{j-1} . This process is continued till all the N image pixels are processed. When the complete image has been processed, the pseudo-projection of the image is distributed in the local memories of the N_p processors in the linear array with N_t/N_p pseudo-projection in PE_j . The complexity of the binary search for the average case and the worst case is the same and is given by $O(\log_2 n)$ where n is the number of entries in the table. The binary search is quite simple to implement and is quite efficient.

Each PE_j in the array computes $\Delta P(j)$, the ratio of the difference between the projection data (actual measurement data) and the pseudo-projection data (computed value) on tube j to the projection data on tube j . $\Delta P(j)$ gives the extent of resemblance between the projection of the object and the projection of the reconstructed object.

For handling the backprojection, the architecture employs the same N_p processors. In the optimal case, with one PE per detector tube, each processor holds ΔP for one detector tube. The host sends the box intensities to the processor array through the FIFO. PE_j receives the box intensity from PE_{j-1} and keeps track of the index of the pixel it currently processes. PE_j uses binary search to find out whether the current pixel lies in tube j . If the current pixel lies in tube j , the corresponding probability value and ΔP are used to compute the correction to the pixel intensity. The partially corrected pixel intensity is passed down the pipeline through the FIFO. PE_j receives the next box intensity data from PE_{j-1} . This process is continued till all the N box intensities have traversed the pipeline. The last PE in the linear array gives out the reconstructed image.

V. IMPLEMENTATION OF THE LINEAR ARRAY

The linear array has been implemented using ADSP 21020 [1] DSP chips. Fig. 3 shows the schematic of a PE. 256 Kwords of program memory and 256 Kwords of data memory are attached to the DSP device through the *program memory* space and *data memory* space respectively. Eight PE's are linearly linked using high-speed first-in-first-out (FIFO) buffers (Am 7200 High Density FIFO 256×9 CMOS Memory). Fig. 4 shows the connection between the PE's. Four Am 7200 devices are connected in width expansion mode to form a 256×36 FIFO buffer. The FIFO buffers are mapped in the data memory address space of the DSP's. PE_j can send data to PE_{j+1} by writing into the $FIFO_{j+1}$. PE_j can receive data from PE_{j-1} by reading the $FIFO_j$.

The Am 7200 CMOS FIFO is a 256×9 dual-port static RAM array and it stores the data written into it in sequential order. The dual-port RAM array has dedicated read and write pointers. The built-in flag logic allows the FIFO to accept and output data asynchronously and simultaneously. The PE_j is connected to one of the ports of the FIFO and is controlled

Fig. 3. Schematic diagram of a PE.

Fig. 4. PE-to-PE FIFO buffer.

by the FIFO control signal *full*, which gives the status of the FIFO. It is asserted when the FIFO is full. Similarly, PE_{j+1} is connected to the other port of the FIFO and is controlled by the control signal *empty*. The bit *empty* is asserted when the FIFO is empty, which indicates that no more reads should be made until PE_j writes into the FIFO. The *full* and *empty* control signals of the FIFO are in turn connected to the FLAG.1 and FLAG.0 of the DSP device respectively. These two DSP flag bits are programmed as inputs and are tested through program instructions. Thus the FIFO allows the linear array to operate asynchronously. During the forward step, PE_j receives the pixel_value from PE_{j-1} and transmits the same to PE_{j+1} . PE_j starts processing the pixel. The computational load at each of the PE's is not the same. So PE_j may take a little longer to process a pixel. But as and when it completes, it picks up the next pixel_value and starts processing.

During the backward step, a pixel_value arrives at a PE_j . The PE_j processes the pixel, updates the pixel_value depending upon the contribution from the tube j and then transmits the result to PE_{j+1} . The PE_{j+1} waits till the *empty* signal changes state, which indicates that new pixel_value has arrived.

A typical DSP such as ADSP 21020 has three independent computation units: an arithmetic and logic unit (ALU), a multiplier, and a shifter. The computation units perform single cycle operations. The three units are connected in parallel and they operate in parallel. In a multi-function instruction, multiple functional units operate in parallel. A 10-port register file is used for transferring data among computation units and data buses, and for storing immediate results.

ADSP 21020 has two independent memories—one for *data* and the other for *program instructions and data*. Two independent address generators (DAG) and a program sequencer supply address for memory access. A program sequencer with a 32-word instruction cache allows the ADSP 21020 to access data from both *data memory* and *program memory* and fetch an instruction. In addition, the DAG's are updated to point to

TABLE I
EXECUTION TIME FOR ONE ITERATION OF
THE EM ALGORITHM (PARTITION-BY-TUBE)

Image size	linear array (1 PE)	linear array (8 PE's)	386/387	IBM 6000 workstation
16*16	11.6 ms	2.48 ms	185 ms	39.38 ms
32*32	91 ms	19.8 ms	1.44 s	309 ms
64*64	753.6 ms	162 ms	12.23 s	2.49 s
128*128	6.97 s	1.41 s	105.43 s	21.48 s

TABLE II
PERFORMANCE OF THE EM ALGORITHM ON
MULTIPLE PROCESSORS (IMAGE SIZE: 64*64)

No of processors	1	4	8	12	16
Execution time	753.6 ms	323.7	162 ms	107.8ms	87.5 ms

the next operands. So all these five operations are executed in the same clock cycle. In addition, the PE has a zero overhead loop facility with a single cycle setup and exit.

ADSP 21020 has an integer multiply-accumulate unit. But it lacks a floating point multiply-accumulate unit. A close look at the EM algorithm shows that most of the computation time is spent in executing *multiply* and *multiply-accumulate* type of instructions, and thus a floating point MAC would have still further speeded up the EM algorithm execution.

In the execution of the EM algorithm, the array pixel_val[] and floating point array pixel_tube_probability[] are stored in data memory and program memory, respectively. This allows the PE to read two operands, one from data memory and the other from the program memory simultaneously. The linear array is connected to a PC/386 system to use the resources such as memory, keyboard, display and the disk storage. The ADSP 21020 operates at 33 MHz with a cycle time of 33 ns. The fast DSP memories are made up of 20-ns static memories.

Table I gives the execution time for one iteration of the EM algorithm on a linear array having a single PE, on a linear array with 8 PE's, on a 386/387 running at 33 MHz under Unix operating system and on an IBM 6000 RISC workstation for the partition-by-tube scheme.

The results show that the computational speed of a multi-processor system for the EM image reconstruction algorithm is about 15.5 times better than that of an IBM 6000 RISC workstation. The speed-up of the linear array with 8 nodes is approximately 5. The EM algorithm was executed on a linear array with 4, 8, 12, and 16 nodes for an image size of 64*64. Table II shows the performance of the EM algorithm on multiple processors. The execution time decreases almost linearly with the increase in number of nodes. The total number of detector tubes processed is constant for a fixed image size. The number of detector tubes processed in each of the nodes decreases linearly with an increase in number of nodes. In addition, the memory required to store the data at the nodes reduces linearly with an increase in number of PE's.

The algorithm can also be implemented on a linear array of fixed point DSP devices which are available for as low as \$10 per chip.

VI. CONCLUSIONS

In this paper, we have described the parallelization of the EM algorithm on an linear array topology. The linear

array topology is expandable with more number of PE's. In this current study, a FIFO buffer between the stages has been used. But with more advanced devices such as an ADSP 21060 containing ADSP 21020 core, 4 Mb of static memory with cross bar switch buses, two serial ports, six 4-b link ports and a powerful DMA processor, it is possible to have glueless interconnection between the processors to build a large network. The architecture is not dependent on the DSP chip chosen, and the substitution of the latest DSP chip is straightforward and could yield better speed performance. The EM algorithm breaks down to a sequence of *multiply* and *multiply/accumulate* type of instructions. Since DSP chips are optimized processors for executing *multiply* and *multiply/accumulate* instructions, they give high performance. It has been found that the computational speed performance of the 8-node linear array executing the EM image reconstruction algorithm is comparable to that of IBM 6000 RISC workstation.

ACKNOWLEDGMENT

The authors would like to thank the reviewers for their valuable comments and suggestions.

REFERENCES

- [1] *Analog Devices's Users Manual*, ADSP 21020, 1992.
- [2] C. M. Chen and S. Y. Lee, "Parallelization of the EM algorithm for 3-D PET image reconstruction," *IEEE Trans. Med. Imag.*, vol. 10, no. 4, pp. 513-522, Dec. 1991.
- [3] ———, "On parallelizing the EM algorithm for PET image reconstruction," *IEEE Trans. Parallel Dist. Syst.*, vol. 5, no. 8, pp. 860-873, Aug. 1994.
- [4] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *J. R. Stat. Soc. Series B*, vol. 39, pp. 1-38, 1977.
- [5] E. Shieh *et al.*, "High-speed computation of the radon transform and back projection using an expandable multi processor architecture," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 2, no. 4, pp. 347-359, Dec. 1992.
- [6] R. Hartz, D. Bristow, and N. Mullani, "A real-time TOFPET slice back projection engine employing dual AM 29116 microprocessors," *IEEE Trans. Nucl. Sci.*, vol. NS-32, no. 1, pp. 839-842, Feb. 1985.
- [7] T. Hebert and R. Leahy, "Fast methods for incorporating attenuation in the EM algorithm," *IEEE Trans. Nucl. Sci.*, vol. 37, no. 2, pp. 754-758, 1990.
- [8] W. F. Jones, L. G. Byars, and M. E. Casey, "Design of a super fast three-dimensional projection system for positron emission tomography," *IEEE Trans. Nucl. Sci.*, vol. 35, no. 2, pp. 800-804, Apr. 1990.
- [9] ———, "Positron emission tomographic and expectation maximization: A VLSI architecture for multiple iterations per second," *IEEE Trans. Nucl. Sci.*, vol. 35, no. 1, pp. 620-624, Feb. 1988.
- [10] L. Kaufman, "Implementing and accelerating the EM algorithm for PET," *IEEE Trans. Med. Imag.*, vol. 6, pp. 37-50, Mar. 1987.
- [11] K. Lange and R. Carson, "EM reconstruction algorithm for emission and transmission tomography," *J. Comput. Assisted Tomography*, vol. 8, pp. 306-316, 1984.
- [12] K. Lange, M. Bahn, and R. Little, "A theoretical study of some maximum likelihood algorithm for emission and transmission tomography," *IEEE Trans. Med. Imag.*, vol. MI-6, no. 2, pp. 106-114, 1987.
- [13] R. M. Lewitt and G. Muehllehner, "Accelerated iterative reconstruction for positron emission tomography based on EM algorithm for maximum likelihood estimation," *IEEE Trans. Med. Imag.*, vol. MI-5, no. 1, pp. 16-22, 1986.
- [14] K. Rajan, L. M. Patnaik, and J. Ramakrishna, "High-speed computation of the EM algorithm for PET image reconstruction," *IEEE Trans. Nucl. Sci.*, vol. 41, no. 5, Oct. 1994.
- [15] N. Rajcevan, K. Rajagopal, and G. Krishna, "Vector-extrapolated fast maximum likelihood estimation algorithms for emission tomography," *IEEE Trans. Med. Imag.*, vol. 11, no. 1, pp. 9-20, Mar. 1992.
- [16] L. A. Shepp and Y. Vardi, "Maximum likelihood reconstruction for emission tomography," *IEEE Trans. Med. Imag.*, vol. 11, no. 2, pp. 113-121, Oct. 1992.
- [17] C. J. Thompson and T. M. Peters, "A fractional address accumulator for fast back projection," *IEEE Trans. Nucl. Sci.*, vol. NS-28, no. 4, pp. 3648-3650, Aug. 1981.
- [18] E. Veclerov and J. Llacer, "Stopping rule for the MLE algorithm based on statistical hypothesis testing," *IEEE Trans. Med. Imag.*, vol. MI-6, pp. 313-319, 1987.