FAIR: A Hardware Architecture for Real-Time 3-D Image Registration

Carlos R. Castro-Pareja, *Member, IEEE*, Jogikal M. Jagadeesh, *Member, IEEE*, and Raj Shekhar, *Member, IEEE*

***Abstract*—Mutual information-based image registration, shown to be effective in registering a range of medical images, is a computationally expensive process, with a typical execution time on the order of minutes on a modern single-processor computer. Accelerated execution of this process promises to enhance efficiency and therefore promote routine use of image registration clinically. This paper presents details of a hardware architecture for real-time three-dimensional (3-D) image registration. Real-time performance can be achieved by setting up a network of processing units, each with three independent memory buses: one each for the two image memories and one for the mutual histogram memory. Memory access parallelization and pipelining, by design, allow each processing unit to be 25 times faster than a processor with the same bus speed, when calculating mutual information using partial volume interpolation. Our architecture provides superior per-processor performance at a lower cost compared to a parallel supercomputer.**

***Index Terms*—Biomedical image processing, digital systems, image registration, pipeline processing.**

I. Introduction

MEDICAL image registration is the process of aligning two or more images that represent the same anatomy at different times, from different viewing angles or using different sensors. These images can be of either the same subject or different subjects. Image registration in medical imaging is used to merge or compare images obtained from a variety of modalities, such as magnetic resonance imaging (MRI), computed tomography (CT), positron emission tomography (PET), single photon emission computed tomography (SPECT), and ultrasound. Common medical applications of image registration are multimodality fusion of anatomical (CT or MRI) and functional (PET or SPECT) images for accurate localization of active tumors, as well as delineation of their shape and size [1]–[3], registration of serial images for monitoring the progression or regression of a disease [4], and postoperative follow-up [5], and brain atlas registration, in which a brain image of a given patient is morphed into a predefined template to identify and label specific regions of the brain [6], [7]. In general, the registration of single-modality images allows monitoring changes over time, whereas the registration of multimodality images combines the complementary structural and functional information about a certain organ.

There are many approaches to three-dimensional (3-D) image registration [8]. To justify hardware implementation, it is imperative that the selected registration algorithm be as general and as flexible as possible. Moreover, this algorithm should be accurate and robust and must not require manual interaction. Voxel similarity-based algorithms fulfill the above criteria better than feature-based approaches [9]. In this paper, we present the hardware implementation of an algorithm that uses the mutual information measure of voxel similarity. Mutual information-based image registration can be fully automatic and applicable to single- or multimodality images of most organs and supports both rigid and nonrigid transformation modes. Most importantly, it is one of the most reliable, robust and promising methods currently available [9]–[12].

Mutual information-based image registration relies on the maximization of the mutual information between two images. Mutual information is a function of two 3-D images and a transformation between them. The 4×4 transformation matrix contains information about rotation, translation, scaling, and shear, in the most general case. Mutual information-based registration uses an optimization algorithm that searches for the transformation matrix that orients the two images such that the mutual information between them is maximized. Powell's method, the downhill Simplex method and simulated annealing are the optimization algorithms commonly employed for the task [13].

Mutual information-based 3-D image registration is an automatic but computationally intensive task, whose typical execution time is on the order of minutes on most modern desktop computers [14], [15]. The total execution time can easily exceed an hour when registering 3-D cardiac image sequences (10–30 images per sequence), an emerging image registration application [16]. Previous attempts to accelerate image registration by using parallel supercomputers [14], [17], [18] achieved significant speed increases, but with a speedup-per-processor ratio smaller than one. Due to communication delays, this ratio tends to decrease as the number of processors increases. Rohlfing *et al.* [14] report speedup-per-processor ratios between 1.00 and 0.32 for single and 64 processor configurations, respectively. While such research is very valuable in understanding parallelism, general purpose supercomputers are expensive and usually have limited availability for applications in clinical environments.

Manuscript received June 2, 2003; revised September 6, 2003 and September 15, 2003. This work was supported by the Department of Defense Research Grant DAMD17-99-1-9034.

C. R. Castro-Pareja is with the Department of Electrical Engineering, The Ohio State University, Columbus, OH 43210 USA, and also with the Department of Biomedical Engineering, Lerner Research Institute, The Cleveland Clinic Foundation, Cleveland, OH 44195 USA.

J. M. Jagadeesh is with the Department of Electrical Engineering and the College of Pharmacy, The Ohio State University, Columbus, OH 43210 USA.

R. Shekhar is with the Department of Biomedical Engineering, Lerner Research Institute, The Cleveland Clinic Foundation, Cleveland, OH 44195 USA (e-mail: shekhar@bme.ri.ccf.org).

Digital Object Identifier 10.1109/TITB.2003.821370

Our research focused on accelerating the calculation of mutual information by analyzing the main speed bottlenecks and overcoming them by developing an optimized hardware architecture for efficient calculation of the mutual information, with the goal of achieving registration times on the order of a second, using fewer processors than necessary when using microprocessor-based computers. Mutual information calculation is a memory-intensive task that does not fully benefit from cache-based memory architectures present in most modern computers. Our novel Fast Automatic Image Registration (FAIR) architecture for hardware-accelerated automatic image registration, presented in this article, enables 3-D image registration at speeds of at least one order of magnitude above the fastest CPU-only software implementation with a higher speedup-per-processor ratio than with standard parallel computers. The improved speedup-per-processor ratio results from a custom pipeline, which reduces the serial component of the algorithm by employing parallel memory access techniques previously used in volume rendering hardware to increase the voxel access rate [19]. As with parallel computers, distributed processing can be used to further enhance the processing speed. Having a higher speedup-per-processor ratio allows us to achieve real-time registration with fewer processing units. Our solution results in a physically smaller, more economical and highly scalable system, which, we believe, will promote a wider use of medical image registration and extend its use into new areas such as four-dimensional (4-D) (3-D space + time) cardiac image registration [20] and image-guided surgery. In the latter application, intraoperative images, such as those obtained from a real-time 3-D ultrasound or MRI scanner, can be used to warp (update) the high-resolution preoperative MRI or CT images [21]–[25].

II. Algorithm

A. Registration by Maximization of Mutual Information

Image registration by maximization of mutual information was introduced by Wells *et al.* [11]. The method attempts to find the transformation $\hat{T}$ that best aligns a reference image RI, with coordinates x, y, and z, and a floating image FI.

$$\hat{T} = \arg\max_{T} MI\left(RI(x,y,z), FI\left(T(x,y,z)\right)\right). \quad (2.1)$$

Mutual information is calculated from individual and joint entropies using the following equation

$$MI(RI,FI) = H(RI) + H(FI) - H(RI,FI). \quad (2.2)$$

The individual entropies $H(RI)$ and $H(FI)$ and the joint entropy $H(RI,FI)$ are computed as follows:

$$H(RI) = -\sum_{a} p_{RI}(a) \ln p_{RI}(a) \quad (2.3)$$

$$H(FI) = -\sum_{b} p_{FI}(b) \ln p_{FI}(b) \quad (2.4)$$

$$H(RI,FI) = -\sum_{a,b} p_{RI,FI}(a,b) \ln p_{RI,FI}(a,b). \quad (2.5)$$

The joint voxel intensity probability $p_{RI,FI}(a,b)$, i.e., the probability of a voxel in the reference image having an intensity a given that the corresponding voxel in the floating image has an intensity b, can be obtained from the joint or mutual histogram of the two images. The mutual histogram represents the joint intensity probability distribution. In the process of mutual information-based registration, the dispersion of values within the mutual histogram is minimized, which in turn minimizes the joint entropy and maximizes the mutual information.

Calculation of the mutual information can be divided into two steps. The first step is to compute the mutual histogram. In the second step, both individual and joint entropies are calculated from the mutual histogram data. These entropies are then used to obtain the mutual information as per (2.2). Using mutual histogram to compute individual entropies also ensures that only those voxels inside the volume of overlap figure in the mutual information calculation.

Since the transformed location of a voxel of the floating image may not coincide with the location of a voxel in the reference image, interpolation is needed. Interpolation also helps in obtaining subvoxel accuracy. Typical interpolation algorithms include nearest neighbor, trilinear interpolation and partial volume interpolation. Partial volume interpolation, as suggested by Maes *et al.* [12], is used to map the voxels in the reference image to their corresponding locations in the floating image. Nearest neighbor interpolation was not considered because it does not provide subvoxel accuracy. Both trilinear interpolation and partial volume interpolation provide subvoxel accuracy. However, Maes *et al.* [12] showed that trilinear interpolation typically introduces new intensity levels as a result of interpolation, causing unpredictable variations in mutual histogram values as the transformation matrix changes. On the other hand, partial volume interpolation accumulates the eight interpolation weights directly into the mutual histogram instead of calculating a resulting intensity level and incrementing that intensity level's mutual histogram count by one, as in trilinear interpolation. Therefore, the main advantage of partial volume over trilinear interpolation is that it produces a mutual histogram, whose values change smoothly with small changes in the transformation, thus resulting in a smoother mutual information surface. Fig. 1 shows a comparison between calculating the mutual histogram using trilinear interpolation and partial volume interpolation. Capek *et al.* [10] showed the difference in mutual information surface smoothness when using different interpolation schemes for mutual information or generalized mutual information (a slight variant of mutual information) calculation. The authors concluded that mutual information, computed according to Maes, provides the smoothest mutual information surface among statistical voxel similarity measures.

B. Transformation Calculation

The transformation that maps the floating image to the reference image is defined by a 4×4 transformation matrix T. The transformation matrix contains information about the rotation, translation, scaling and shear parameters that are inherent to the

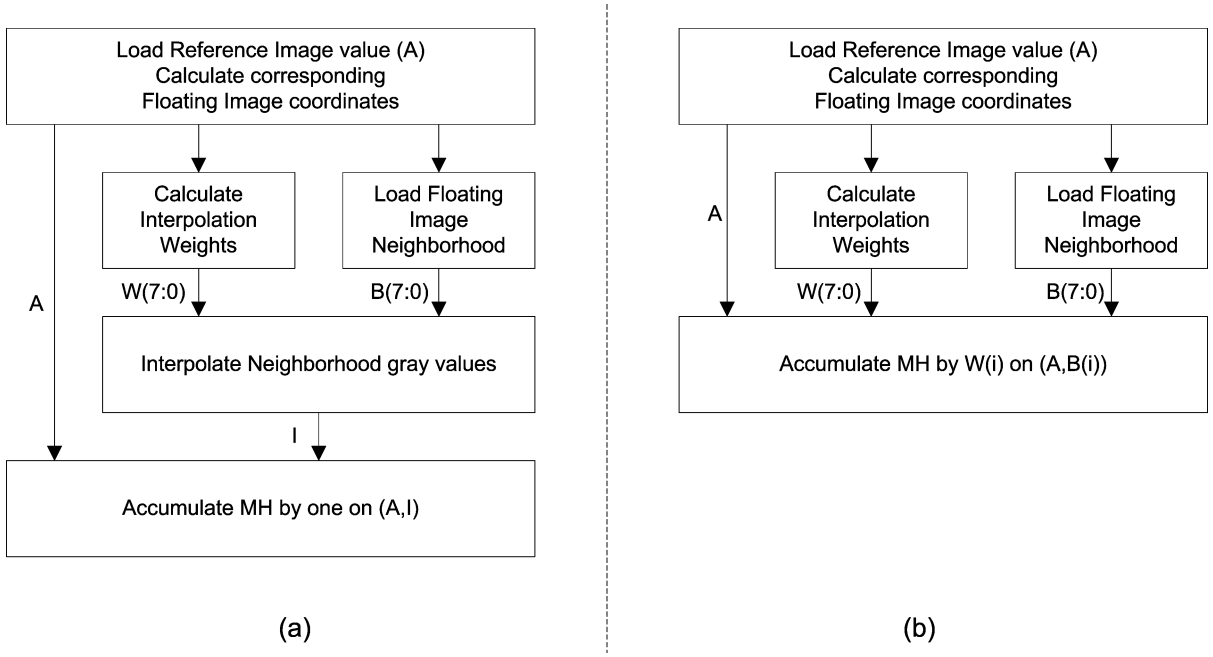

Fig. 1. Mutual histogram calculation using (a) trilinear interpolation and (b) partial volume interpolation.

transformation being performed. The scaling parameters also incorporate voxel scaling necessary to compare images with different voxel sizes. Voxel scaling factors are constant for rigid registration and as such are excluded from optimization. Since the inverse transformation (T^{-1}) is needed to generate and visualize the transformed image, the algorithm tries to estimate it instead of the direct transformation. The location of a given reference image voxel $\overline{v}_r$ in the floating image is given by

$$\overline{v}_f = T^{-1} \cdot \overline{v}_r. \tag{2.6}$$

Since the elements of T^1 have both integer and fractional components, the transformed location of a reference image voxel $\overline{v}_r$ also has an integer component $\overline{v}_{fint} = floor(\overline{v}_f)$ and a fractional component $\overline{v}_{ffroc} = \overline{v}_f \bmod 1$. The integer component is used to obtain the base memory address of the corresponding $2 \times 2 \times 2$ neighborhood in the floating image. According to the partial volume interpolation algorithm, the interpolation weights are calculated from the fractional components, which are accumulated into the mutual histogram at the coordinates set by their corresponding reference image and floating image values. Equations (2.7)–(2.14) show the partial volume interpolation calculation process, where $\hat{x}$, $\hat{y}$ and $\hat{z}$ are unit vectors in the x, y, and z directions, respectively.

$$MH\,(RI(\overline{v}_r), FI(\overline{v}_{fint})) + = (1 - v_{ffracx}) \cdot (1 - v_{ffracy}) \cdot (1 - v_{ffracz}) \tag{2.7}$$

$$MH\,(RI(\overline{v}_r), FI(\overline{v}_{fint}\hat{x})) + = v_{ffracx} \cdot (1 - v_{ffracy}) \cdot (1 - v_{ffracz}) \tag{2.8}$$

$$MH\,(RI(\overline{v}_r), FI(\overline{v}_{fint} + \hat{y})) + = (1 - v_{ffracx}) \cdot v_{ffracy} \cdot (1 - v_{ffracz}) \tag{2.9}$$

$$MH\,(RI(\overline{v}_r), FI(\overline{v}_{fint} + \hat{z})) + = (1 - v_{ffracx}) \cdot (1 - v_{ffracy}) \cdot v_{ffracz} \tag{2.10}$$

$$MH\,(RI(\overline{v}_r), FI(\overline{v}_{fint} + \hat{x} + \hat{y})) + = v_{ffracx} \cdot v_{ffracy} \cdot (1 - v_{ffracz}) \tag{2.11}$$

$$MH\,(RI(\overline{v}_r), FI(\overline{v}_{fint} + \hat{x} + \hat{z})) + = v_{ffracx} \cdot (1 - v_{ffracy}) \cdot v_{ffracz} \tag{2.12}$$

$$MH\,(RI(\overline{v}_r), FI(\overline{v}_{fint} + \hat{y} + \hat{z})) + = (1 - v_{ffracx}) \cdot v_{ffracy} \cdot v_{ffracz} \tag{2.13}$$

$$MH\,(RI(\overline{v}_r), FI(\overline{v}_{fint} + \hat{x} + \hat{y} + \hat{z})) + = v_{ffracx} \cdot v_{ffracy} \cdot v_{ffracz}. \tag{2.14}$$

The process of accumulating the interpolation weights for a voxel in the floating image into the mutual histogram can be divided into three steps: (a) $\overline{v}_f$ calculation as per (2.6), (b) calculation of interpolation weights [right-hand sides (RHS) of (2.7)–(2.14)], and (c) accumulation of interpolation weights into the mutual histogram as per (2.7)–(2.14).

C. Complexity Analysis

Mutual information-based registration usually requires hundreds of iterations (mutual information evaluations), depending on the optimization algorithm used to maximize the mutual information function, the image complexity, and the degree of initial misalignment. If real-time performance (in the order of a second) is required, the mutual information calculation time should be on the order of tens of milliseconds.

Constructing the mutual histogram, the first step in mutual information calculation, involves performing partial volume interpolation n times, where n is less than or equal to the number of voxels in the reference image. The number of operations in the second step, the calculation of mutual information as per (2.2), is a function of the size of the mutual histogram. Since the number of bins in the mutual histogram is usually an order of magnitude smaller than n, it is the calculation of the mutual histogram that dominates the mutual information calculation time.

Table I shows the fractions of the total processing time spent on mutual histogram calculation and the remaining computations for different image sizes. Mutual histogram calculation consumes approximately 99% or more of the total computation time for most practical medical images, and its computational share varies only slightly depending on the specific algorithm implementation and the processor, memory architecture, compiler and operating system used to run the algorithm.

Since the majority of the registration execution time is spent on calculating the mutual histogram, accelerating mutual histogram calculation has been the focus of our work. Our analysis shows that the partial volume interpolation is the primary performance bottleneck in mutual histogram calculation. At current microprocessor speeds, the time of mutual histogram calculation for 3-D images is dictated almost exclusively by the memory access time. From (2.7)–(2.14), 25 memory accesses are needed to perform partial volume interpolation per voxel of the reference image: 1 to access the reference image voxel $(\overline{v}_r)$, 8 to access the 8-voxel neighborhood in the floating image and 16 accesses to the mutual histogram memory (8 reads and 8 writes). Accesses to the reference image are sequential and benefit from standard caching techniques. The mutual histogram memory has a small size (256×256 or 64 K values) and thus accesses to it have high locality of reference. However, the floating image is accessed in a direction across the image that depends on the transformation being applied. Unless there is no rotation component, this direction is not parallel to the direction in which voxels are stored, hence accesses have poor locality and do not benefit from memory-burst accesses or memory-caching schemes. Accesses to the floating image therefore depend almost exclusively on the memory bus speed. Since memory access time does not evolve according to Moore's law, mutual information-based registration times are not expected to be significantly reduced in the near future by enhanced single-processor computer architectures.

D. Impact of Accelerating Mutual Histogram Calculation

The speedup in registration achieved by accelerating the mutual histogram calculation depends on the share of the overall registration execution spent on calculating the mutual histogram. Equation (2.15) shows Amdahl's law [26], which gives the resulting overall speedup for a process when a part of it is accelerated. The serial part f_serial corresponds to the proportion of the overall process execution time that is not being accelerated, while the parallel part f_parallel corresponds to the proportion $(1 - f\_\text{serial})$ that is being accelerated.

$$\text{Speedup} = \left(f\_\text{serial} + \left(\frac{f\_\text{parallel}}{\text{Speedup\_parallel}} \right) \right)^{-1}. \quad (2.15)$$

In our case, the serial part corresponds to the time spent on the optimization algorithm and on the accumulation of logarithms performed to obtain the mutual information value from the mutual histogram (2.2)–(2.5), while the parallel part corresponds to the mutual histogram calculation. To determine the effective registration speedup resulting from accelerating mutual histogram calculation, we ran several experiments in which we obtained the total registration time and the time spent on mutual histogram calculations. Mutual histogram calculation time depends on the image size and is shown as a fraction of the total calculation time in Table I. Since mutual histogram calculation time depends on the number of voxels in the image, the maximum speedup predicted by Amdahl's law also depends on the number of voxels. Typical 3-D medical images are approximately 2^{18} to 2^{24} voxels large. For this range, the expected maximum registration speedup is approximately between 90 and 3000, when the execution time of the parallel part becomes negligible compared with the execution time of the serial part. The minimum calculation time achieved by this speedup is constant for a given dataset, and it is equivalent to the time the computer spends on calculating mutual information from the mutual histogram (accumulation of logarithms) for all iterations leading up to the optimal transformation and executing the optimization algorithm. The majority of this time is spent on computing logarithms, which on average takes 10–30 ms per iteration on most modern computers. In comparison, the time spent on the optimization algorithm is negligible (less than 0.1 ms). Considering that a complete registration usually requires hundreds of iterations, the acceleration of mutual histogram calculation can reduce registration time to no more than a few seconds.

TABLE I
MUTUAL HISTOGRAM CALCULATION COMPONENT PER IMAGE SIZE

Image Size (voxels)	Mutual Histogram Calculation Component	Serial Component
64x64x64 (2^{18})	98.90%	1.10%
128x128x128 (2^{21})	99.84%	0.16%
256x256x256 (2^{24})	99.97%	0.03%

III. MUTUAL HISTOGRAM CALCULATION: SIMILARITIES WITH VOLUME RENDERING

Mutual histogram calculation has many similarities with the ray casting algorithm used for volume rendering. In both cases, a 3-D image is traversed by casting rays through the 3-D dataset and performing interpolation to obtain equally spaced samples along the ray. In the case of mutual histogram calculation, a second volume is traversed too. The reference volume is traversed by casting rays parallel to the $\hat{x}$ axis, coinciding with the data rows. These same rays are cast through the floating images; however, they may start and end either inside or outside the volume of the floating image, depending on the characteristics of the transformation being applied (rotation, translation, scaling, shearing). Fig. 2 shows an example of a set of rays being cast through the reference image and their possible corresponding accesses in the floating image. Both volume rendering and mutual histogram calculation try to condense 3-D information into a 2-D matrix: the display buffer in the case of volume rendering, and the mutual histogram matrix in the case of mutual histogram calculation. A difference is that volume rendering employs trilinear interpolation, which provides acceptable results for volume visualization, while mutual histogram calculation, as presented here, makes use of partial volume interpolation. This difference changes the focus

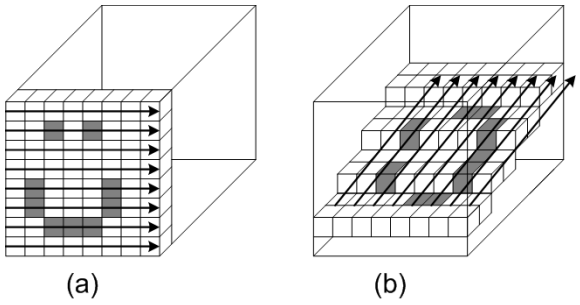

Fig. 2. Ray casting through (a) the reference image and (b) through the floating image .

of optimization on how volume memory is accessed. In volume rendering, the major bottleneck in the interpolation pipeline is volume memory access, which is why it is highly desirable to obtain near static RAM (SRAM) performance for the volume memory. This is solved by using custom memory-addressing and memory-caching techniques [19], [27]–[29] that permit parallel access of full $2 \times 2 \times 2$ voxel neighborhood (useful for interpolation) and/or allowing the parallel access of whole rays along the volume (for compositing). In the case of mutual histogram calculation, a similar bottleneck exists when accessing the floating image. Because the reference image is accessed sequentially, it can be performed efficiently using memory-burst transfers. On the other hand, the number of accesses to the mutual histogram RAM required for each interpolated voxel is 16 times higher because of the use of partial volume interpolation. Therefore, the performance bottlenecks in mutual histogram calculation are in accessing floating image and mutual histogram memories.

IV. FAIR Architecture

The FAIR architecture is designed to accelerate mutual histogram calculation by accelerating partial volume interpolation. As described in Sections II-B and II-C, partial volume interpolation is performed by accessing one reference image voxel at a time, calculating the coordinates of the corresponding floating image voxel neighborhood, accessing the floating image voxel locations of the eight elements of the neighborhood, calculating the corresponding interpolation weights and finally updating the mutual histogram. These tasks need to be repeated for each voxel in the reference image. In a standard (single processor) software implementation of the algorithm, these tasks must be executed before processing the next voxel, which means that the total mutual histogram calculation time equals the product of single voxel interpolation time and the number of voxels in the reference image. The FAIR architecture optimizes partial volume interpolation by means of pipelining, parallel memory access, and distributed processing.

A. Pipeline

The first level of algorithm parallelization comes from pipelined execution. The FAIR architecture uses the 3-stage pipeline shown in Fig. 3. This arrangement takes advantage of the independence between the three partial volume interpolation tasks listed in Section II-B. With pipelined execution, the total time of mutual histogram calculation becomes equal to the average pipeline stage calculation time times the number of voxels in the reference image.

The first stage accesses the reference image memory sequentially and calculates the coordinates of the corresponding floating image voxels. The integer part of the coordinates is passed to the floating image RAM controller as the floating image address, while the fractional part is passed to the interpolator. The second stage calculates the interpolation weights and accesses the floating image. The third stage accumulates the interpolation weights into the mutual histogram at the positions given by the reference image value and the corresponding eight floating image voxel values.

B. Parallel Memory Access Scheme

Each pipeline stage must have its own memory bus for parallel execution. The first and second stage memory buses are used for the reference image and floating image memory accesses, respectively, while the third stage memory bus is used for mutual histogram memory accesses. The three memory buses differ in size, speed and access requirements. The reference image memory is accessed sequentially, while the floating image and mutual histogram memories are accessed randomly (i.e., nonsequentially). The reference image and floating image memories have a size of 16 M $\times$ 9 each and the mutual histogram memory has a size of 64 K $\times$ 32, corresponding to a mutual histogram of size 256×256. Since a pipeline's stage latency equals the latency of the slowest stage, it is important to minimize the memory access time of those stages that require more than one access per voxel.

The mutual histogram memory has the most stringent access speed requirements since it needs to be accessed 16 times per interpolation. Because the mutual histogram memory is small, it can be implemented using high-speed SRAM, which is at least an order of magnitude faster than the dynamic RAMs used for the two images.

Because the 3-D images are large, the use of high-speed SRAM is currently not cost effective for implementing image memories. Between the two images, the reference image has more relaxed requirements, since it is accessed sequentially (in an x-y-z order) to perform interpolation. This kind of access benefits from burst accesses and memory caching techniques, making the use of a single Synchronous Dynamic RAM (SDRAM) bus for image storage a viable option. On the other hand, the floating image must be accessed randomly and has to provide eight voxel values per reference image voxel, which calls for a different way to store the floating image data.

To overcome the floating image memory access bottleneck, the FAIR architecture employs memory-addressing techniques similar to the ones used in volume rendering hardware to speed up interpolation. Doggett and Meißner [19] presented a parallel memory addressing scheme called Cubic Addressing. The main advantage of Cubic Addressing is that it enables parallel access to a $2 \times 2 \times 2$ voxel neighborhood, thus greatly reducing the number of sequential memory accesses otherwise needed for interpolation. It uses eight parallel memory modules and a custom address decoder to calculate the voxel addresses.

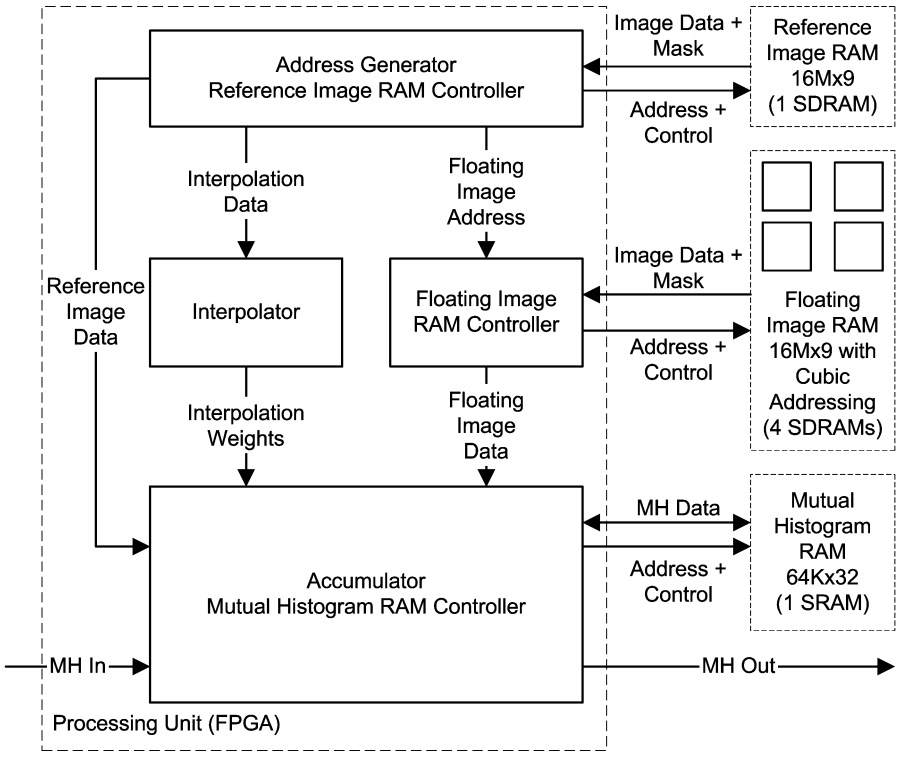

Fig. 3. A FAIR processing unit.

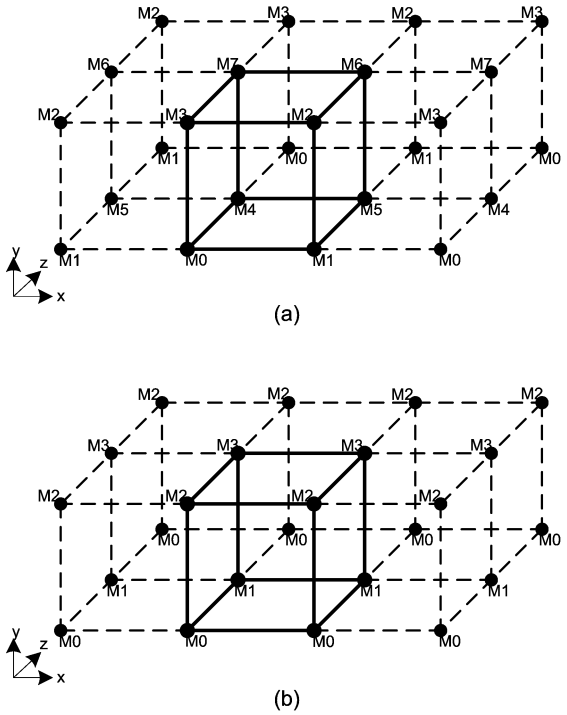

Fig. 4. Memory module assignments of voxels in Cubic Addressing. (a) Original scheme described in [19]. (b) Burst-access cubic addressing implemented in the FAIR architecture.

This memory arrangement is shown in Fig. 4(a). A similar addressing scheme that employs only four parallel memory modules [Fig. 4(b)] is used in the FAIR architecture to access the floating image memory. This scheme takes advantage of the fast burst-mode access of SDRAMs to reduce the number of necessary parallel memory modules. Size-2 burst accesses take the same amount of time as single accesses on SDRAMs. By storing neighboring voxels sequentially along the x direction, a size-2 burst access to the four memory modules, starting on the neighborhood voxels with the lowest x coordinate, will retrieve information about two adjacent 2×2 y-z-plane neighborhoods. A caching scheme here would not enhance performance because the SDRAM random access time is comparable to the time required to perform the 16 mutual histogram SRAM accesses.

C. Distributed Processing

Mutual histogram calculation lends itself well to parallelization by dividing up the reference image into a number of nonoverlapping subvolumes and distributing them among an equal number of processing units, each with its own mutual histogram calculation pipeline [14]. Each processing unit has the necessary RAM to store its part of the reference image, the full floating image and the mutual histogram. A full copy of the floating image is needed at each processing unit because the portion of the floating image that matches the reference image subvolume depends on the transformation and can be located anywhere inside the floating image (Fig. 5). Each processing unit will calculate the partial mutual histogram corresponding to its subvolume. When all processing units are finished computing, the full mutual histogram can be obtained by adding the partial mutual histograms. As shown in Fig. 3, each processing unit has an input port to add the partial mutual

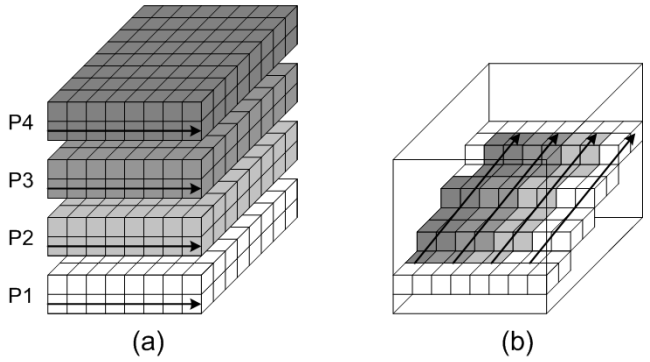

Fig. 5. (a) Division of the reference image into four subvolumes to be assigned to four individual FAIR processing units for distributed computing. (b) Possible gray level-coded floating image regions corresponding to reference image subvolumes.

histogram of the previous unit and an output port to transmit the result to the next unit or the host computer. The resulting mutual histogram calculation time can be obtained as

$$T_{MH} = t_{\text{vox}} \cdot \frac{n}{p} + t_{\text{tran}} \cdot m^2 + t_{\text{latency}} \tag{4.1}$$

where t_{vox} is the calculation time per voxel (equal to the pipeline stage latency); n, the number of voxels in the reference image; m^2, the size of the mutual histogram matrix (usually 2^{16} bins); t_{tran}, the time required to transmit one mutual histogram value between two processing units or between a processing unit and the host computer and p, the number of processing units. The first term in the T_{MH} formula is the time it takes for each unit to calculate its share of the mutual histogram. The second term corresponds to the time spent on transmitting the mutual histogram to the host computer. The third latency term, which is negligible, accounts for the time required to fill the mutual histogram calculation pipelines and the partial mutual histograms addition pipeline. Given that interpolation of a voxel requires 25 memory accesses, the upper bound for the mutual histogram calculation speedup factor of an implementation with p processing units with respect to a computer (ignoring t_{latency}) is as follows:

$$\text{Speedup} = \frac{f_{PU\_RAM}}{f_{PC\_RAM}} \frac{25 \cdot n}{\left(\frac{n}{p}\right) + m^2 \cdot \left(\frac{t_{tran}}{t_{vox}}\right)} \tag{4.2}$$

where f_{PC_RAM} is the computer's RAM bus clock frequency and f_{PU_RAM} is the processing unit's SDRAM clock frequency. This upper bound is the worst-case scenario and assumes that there are no cache hits when calculating the complete mutual histogram on the computer. Furthermore, the maximum theoretical speedup-per-processor achievable using this architecture is 25, when $f_{PC\_RAM} = f_{PU\_RAM}$ and $n/p \gg m^2 \cdot (t_{tran}/t_{vox})$.

V. Implementation and Results

A. Time

As a proof of concept, the FAIR architecture was implemented in an external prototype board using Altera ACEX 1K100 FPGAs. We implemented two processing units per board, each using two FPGAs. The limiting resource on the FPGAs was the number of I/O pins. Internal resource utilization was about 60%. The image RAMs were implemented using PC100 SDRAMs, and the mutual histogram RAM was implemented using high-speed SRAMs. The board operated at a clock rate of 80 MHz. It was connected to a host computer using a PCI-7200 digital I/O board by ADLINK Technology (Taipei, Taiwan). In Tables II and III, the actual timings are compared with analogous software implementation timings on a 1-GHz Pentium III computer with a 133 MHz memory bus. In our implementation, the $t_{\text{tran}}/t_{\text{vox}}$ ratio was equal to 2.67. The upper limit of the speedup presented in Table III was calculated using (4.2). The size of the mutual histogram was 256×256. The software calculation time depends on the number of cache misses, which in turn depends on the direction in which the floating image is accessed. To reflect the different possible cases, the average software calculation times were obtained by performing a series of mutual information calculations with transformations covering the whole range of image rotation values with 5-degree increments. Our system achieved a significant speedup-per-processor ratio in all cases. Using a faster memory bus (i.e., increasing f_{PU_RAM}) and a faster I/O bus to transmit the mutual histogram to the host computer (i.e., reducing t_{tran}) would increase the speed further.

TABLE II
Comparison of Mutual Information Calculation Time in ms

Implementation	Image Size (voxels)		
	2^{18}	2^{21}	2^{24}
Software (PIII, 1 GHz)	430	3500	28000
FAIR (1 Unit)	62	430	3370
FAIR (2 Units)	36	220	1700

B. Accuracy

Mutual histogram calculation involves transforming the coordinates of the voxels of the reference image to obtain their corresponding locations in the floating image, and performing partial volume interpolation to calculate the weights to be added to the mutual histogram. Typical mutual histogram sizes are between 32×32 and 256×256. Studholme [30] showed that varying the mutual histogram size in this range does not affect the outcome of registration significantly.

The largest possible mutual histogram entry is equal to the size of the reference image—a situation that arises when each image is uniform (has a single intensity). Because 3-D images used in medical applications commonly have on the order of 2^{24} voxels ($256 \times 256 \times 256$ sized image), the smallest word length for the mutual histogram RAM should be 24 bits, for positive integer mutual histogram values. It is also necessary to have support for fractional values inside the mutual histogram since partial volume interpolation provides a set of eight fractional values that are accumulated into the mutual histogram. So any numerical representation used to calculate the mutual histogram should have more than 24 bits in the mantissa. This requirement rules out the use of single-precision floating-point numbers to accumulate and store the mutual histogram, since their mantissa is only 23 bits long. Better alternatives for mutual histogram accumulation and storage are double-precision floating-

TABLE III
MUTUAL INFORMATION SPEEDUP RATES

Implementation	Image Size (voxels)					
	2^{18}		2^{21}		2^{24}	
	Speedup	Upper Bound	Speedup	Upper Bound	Speedup	Upper Bound
Software (PIII, 1 GHz)	1.00	1.00	1.00	1.00	1.00	1.00
FAIR (1 Unit)	6.94	9.02	8.14	13.88	8.31	14.88
FAIR (2 Units)	11.94	12.88	15.91	25.77	16.47	29.46

point numbers and fixed-precision numbers with a word length greater than 24 bits. Double-precision floating-point numbers have a dynamic range that is far in excess of what is necessary to perform mutual histogram calculation since all values are aligned (making the exponent bits unnecessary). Furthermore, implementing floating-point arithmetic requires considerably more resources than fixed-point, so we decided to use a fixed-point representation to accumulate and store the mutual histogram.

In our prototype system, partial volume interpolation was implemented using a 32-bit, fixed-point approach. The system used 8 bits for the fractional part, resulting in an accuracy of 1/256th of a voxel dimension, and 24 for the integer part. To validate our approach, the fixed-point implementation was compared with a C++ implementation using double-precision floating-point accuracy. The rounding effect from the use of fixed-point arithmetic produced an offset error that resulted in mutual information surface being elevated with respect to its analytical version, and a small reduction in the dynamic range of the mutual information values across the mutual information surface by a small factor (less than 5% for 32-bit fixed-point), equivalent to a global linear scaling. These errors neither changed the overall shape of the mutual information surface nor the location of the maximum. The accuracy of registration was therefore not affected. Experiments using both single-modality (MRI against MRI and CT against CT) and multimodality (CT against MRI, PET against MRI, and PET against CT) data sets yielded practically the same results (with errors on the order of 1/100th of a voxel dimension) using hardware-accelerated and software-based registration approaches.

VI. CONCLUSION

Image registration through maximization of mutual information is computationally intensive, demanding execution times on the order of minutes on modern desktop computers. Because this algorithm is memory access limited, continuing rise in the microprocessor speed leads to only moderate increase in the algorithm's speed. To overcome this fundamental computing limitation, we developed a new hardware architecture, called FAIR, for distributed, real-time 3-D image registration.

The FAIR architecture derives its speed from 1) a custom interpolation pipeline with independent memory busses and 2) distributed processing. A practical implementation with standard PC100 SDRAMs, operating at 80 MHz, provided 8.3-fold speed increase compared to a 1-GHz Pentium III computer with PC133 SDRAMs running at 133 MHz. This speedup, even when using just one processing unit, was significant. For additional speed, the modularity of the architecture can be exploited to efficiently implement arrays of processing units using VLSI or FPGAs to perform distributed image registration. The FAIR architecture allows reaching the maximum possible speed predicted by Amdahl's law using far fewer processing units than standard multiprocessor computers. Distributed processing alone can be implemented using multiprocessor computers or computer clusters, but the benefits of using parallel memory access in the pipeline are not gained, thus yielding significantly lower speedup-per-processor ratios, as reported earlier [14]–[18]. Custom processing units are faster, more compact, more power conserving and significantly less expensive than the nodes of a parallel supercomputer, resulting in a smaller and more economical system suitable for clinical use. The cost of our prototype board, housing two units (processors), was approximately two thousand dollars. A comparable speedup can be obtained using a 16-processor parallel computer, but at a cost of tens to hundreds of thousands of dollars. Real-time 3-D image registration made possible by the FAIR architecture may lead to wider adoption of this useful technology in the diagnosis and treatment of human diseases.

REFERENCES

[1] S. M. Larson, C. R. Divgi, and A. M. Scott, "Overview of clinical radioimmunodetection of human tumors," *Cancer*, vol. 73, no. 3, Feb. 1994.

[2] J. G. Rosenman, E. P. Miller, G. Tracton, and T. J. Cullip, "Image registration: an essential part of radiation therapy treatment planning," *Int. J. Radiation Oncol., Biol., Phys.*, vol. 40, no. 1, pp. 197–205, Jan 1998.

[3] T. Nishioka *et al.*, "Image fusion between 18FDG-PET and MRI/CT for radiotherapy planning of oropharyngeal and nasopharyngeal carcinomas," *Int. J. Radiation Oncol., Biol., Phys.*, vol. 53, no. 4, pp. 1051–1057, July 2002.

[4] S. De Santi *et al.*, "Hippocampal formation glucose metabolism and volume losses in MCI and AD," *Neurobiol. Aging*, vol. 22, no. 4, pp. 529–539, July 2001.

[5] C. R. G. Guttmann *et al.*, "Quantitative follow-up of patients with multiple sclerosis using MRI: reproducibility," *J. Magn. Reson. Imag.*, vol. 9, no. 4, pp. 509–518, Apr. 1999.

[6] B. M. Dawant, S. L. Hartmann, and S. Gadamsetty, "Brain atlas deformation in the presence of large space-occupying tumors," in *Proc. Medical Image Computing and Computer-Assisted Intervention, MICCAI'99, Lecture Notes in Computer Science*, vol. 1679, 1999, pp. 589–596.

[7] J. Mazziotta *et al.*, "A probabilistic atlas and reference system for the human brain: International Consortium for Brain Mapping (ICBM)," *Philosophical Trans. Royal Soc. London Series B—Biolog. Sci.*, pp. 1293–1322, Aug. 2001.

[8] J. B. Maintz and M. Viergever, "A survey of medical image registration," *Med. Image Anal.*, vol. 2, no. 1, pp. 1–36, 1998.

[9] M. Holden *et al.*, "Voxel similarity measures for 3-D serial MR brain image registration," *IEEE Trans. Med. Imag.*, vol. 19, pp. 94–102, Feb. 2000.

[10] M. Capek *et al.*, "Robust and fast medical registration of 3D-multi-modality data sets," in *Proc. Medicon 2001—IX Mediterranean Conf. Medical and Biological Engineering and Computing*, 2001, pp. 515–518.
[11] W. M. Wells, P. Viola, H. Atsumi, S. Nakajima, and R. Kikinis, "Multi-modal volume registration by maximization of mutual information," *Med. Image Anal.*, vol. 1, pp. 35–51, 1996.
[12] F. Maes *et al.*, "Multimodality image registration by maximization of mutual information," *IEEE Trans. Med. Imag.*, vol. 16, pp. 187–198, Apr. 1997.
[13] J. P. W. Pluim, J. B. A. Maintz, and M. A. Viergever, "Mutual-information-based registration of medical images: a survey," *IEEE Trans. Med. Imag.*, vol. 22, pp. 986–1004, Aug. 2003.
[14] T. Rohlfing and C. R. Maurer, "Non-rigid image registration in shared-memory multiprocessor environments with application to brains, breasts, and bees," *IEEE Trans. Inform. Technol. Biomed.*, vol. 7, pp. 16–25, Mar. 2003.
[15] T. Netsch, P. Rösch, A. van Muiswinkel, and J. Weese, "Toward real-time multi-modality 3-D medical image registration," in *Proc. Eighth IEEE Int. Conf. Computer Vision, ICCV 2001*, vol. 1, 2001, pp. 718–725.
[16] R. Shekhar, V. Zagrodsky, M. Garcia, and J.JD. Thomas, "3D Stress echocardiography: a novel application based on registration of real-time 3D ultrasound images," in *Proc. Computer Assisted Radiology and Surgery (CARS), 2002*, 2002, pp. 873–878.
[17] S. K. Warfield, F. Jolesz, and R. Kikinis, "A high performance approach to the registration of medical imaging data," *Parallel Computing*, vol. 24, no. 9–10, pp. 1345–1368, 1998.
[18] G. E. Christensen, M. I. Miller, M. W. Vannier, and U. Grenander, "Individualizing neuroanatomical atlases using a massively parallel computer," *IEEE Comput.*, vol. 29, pp. 32–38, Jan. 1996.
[19] M. Doggett and M. Meißner, "A memory addressing and access design for real time volume rendering," in *Proc. 1999 IEEE Int. Symp. Circuits Syst., ISCAS '99*, vol. 4, 1999, pp. 344–347.
[20] R. Shekhar and V. Zagrodsky, "Mutual information-based rigid and nonrigid registration of ultrasound volumes," *IEEE Trans. Med. Imag.*, vol. 21, pp. 9–22, 2002.
[21] I. Kaplan *et al.*, "Real time MRI-ultrasound image guided stereotactic prostate biopsy," *Magnetic Resonance Imaging*, vol. 20, pp. 295–299, 2002.
[22] B. C. Porter *et al.*, "Three-dimensional registration and fusion of ultrasound and MRI using major vessels as fiducial markers," *IEEE Trans. Med. Imag.*, vol. 20, pp. 354–359, Apr. 2001.
[23] B. Brendel, S. Winter, A. Rick, M. Stockheim, and H. Ermert, "Registration of 3D CT and ultrasound datasets of the spine using bone structures," *Computer Aided Surg.*, vol. 7, no. 3, pp. 146–155, 2002.
[24] S. K. Warfield *et al.*, "Real-time registration of volumetric brain MRI by biomechanical simulation of deformation during image guided neurosurgery," *Computing and Visualization in Science*, vol. 5, no. 1, pp. 3–11, July 2002.
[25] A. Roche, X. Pennec, G. Malandain, and N. Ayache, "Rigid registration of 3-D ultrasound with MR images: a new approach combining intensity and gradient information," *IEEE Trans. Med. Imag.*, vol. 20, no. , pp. 1038–1049, Oct. 2001.
[26] J. L. Hennessy, D. A. Patterson, and D. Goldberg, *Computer Architecture: A Quantitative Approach*, 2nd ed. San Francisco, CA: Morgan Kaufman, 1996.
[27] M. De Boer, A. Gröpl, J. Hesser, and R. Männer, "Latency- and hazard-free volume memory architecture for direct volume rendering," in *Proc. Eleventh Eurographics Workshop on Graphics Hardware*, Aug. 1996, pp. 109–119.
[28] G. Knittel, "Verve: voxel engine for real-time visualization and examination," *Computer Graphics Forum*, vol. 12, no. 3, pp. 37–48, Mar. 1993.
[29] H. Pfister and A. Kaufman, "Cube-4-a scalable architecture for real-time volume rendering," in *Proc. Symp. Volume Visualization*, Oct. 1996, pp. 47–54.
[30] A. Studholme, "Measures of 3D medical image alignment," Ph.D. dissertation, Univ. London, London, U.K., 1997.

Carlos R. Castro-Pareja (M'00) received the B.Sc. degree in electrical engineering from the Pontificia Universidad Catolica del Peru, Lima, Peru, in 1999, and the M.Sc. degree in electrical engineering from The Ohio State University, Columbus, in 2001.

He is currently pursuing the Ph.D. degree with the Department of Electrical Engineering, The Ohio State University. His research interests include development and hardware acceleration of image processing algorithms.

Jogikal M. Jagadeesh (M'74) received the B.S. degree from University College, Bangalore, India, the M.S. degree from the Indian Institute of Science, Bangalore, and the Ph.D. degree from The Ohio State University, Columbus, in 1962, 1964, and 1974, respectively.

He is currently an Associate Professor with the Department of Electrical Engineering and the College of Pharmacy, The Ohio State University. His recent research interests include signal processing and pattern analysis of bioelectric potentials, simulation and modeling of biological systems, and medical imaging, particularly high-field magnetic resonance imaging.

Dr. Jagadeesh has served on the Editorial Board of *IEEE Computer*.

Raj Shekhar (M'94) received the B.Tech. degree in electrical engineering from the Indian Institute of Technology, Kanpur, in 1989, the M.S. degree in bioengineering from the Arizona State University, Tempe, in 1991, and the Ph.D. degree in biomedical engineering from The Ohio State University, Columbus, in 1997.

He worked as a Senior Research Engineer for Picker International (now Philips Medical Systems) for two years before joining the Department of Biomedical Engineering, Cleveland Clinic Foundation, Cleveland, OH, in December 1998. His research interests include medical imaging, image processing, and computer graphics, and, currently, he leads research on real-time 3-D ultrasound and multimodality imaging.