Nonrigid Image Registration in Shared-Memory Multiprocessor Environments With Application to Brains, Breasts, and Bees

Torsten Rohlfing and Calvin R. Maurer, Jr., *Member, IEEE*

***Abstract*—One major problem with nonrigid image registration techniques is their high computational cost. Because of this, these methods have found limited application to clinical situations where fast execution is required, e.g., intraoperative imaging. This paper presents a parallel implementation of a nonrigid image registration algorithm. It takes advantage of shared-memory multiprocessor computer architectures using multithreaded programming by partitioning of data and partitioning of tasks, depending on the computational subproblem. For three different biomedical applications (intraoperative brain deformation, contrast-enhanced MR mammography, intersubject brain registration), the scaling behavior of the algorithm is quantitatively analyzed. The method is demonstrated to perform the computation of intra-operative brain deformation in less than a minute using 64 CPUs on a 128-CPU shared-memory supercomputer (SGI Origin 3800). It is shown that its serial component is no more than 2% of the total computation time, allowing a speedup of at least a factor of 50. In most cases, the theoretical limit of the speedup is substantially higher (up to 132-fold in the application examples presented in this paper). The parallel implementation of our algorithm is, therefore, capable of solving nonrigid registration problems with short execution time requirements and may be considered an important step in the application of such techniques to clinically important problems such as the computation of brain deformation during cranial image-guided surgery.**

***Index Terms*—Brain atlas, contrast-enhanced MR mammography, high-performance computing, intersubject registration, intraoperative brain deformation, motion correction, multithreaded computations, nonrigid image registration, parallel performance.**

I. Introduction

IMAGE-TO-IMAGE registration is a common task in biomedical image processing [1]. The problem of registration arises whenever images acquired from different scanners, at different times, or from different subjects need to be combined for analysis or visualization. Several fast, robust, and accurate intensity-based rigid (with or without scaling) image registration algorithms have been reported and validated [2] and are commonly used for applications where a rigid transformation is appropriate. Nonrigid image registration is an active research area. Nonrigid methods are important for applications where the anatomy deforms or changes over time. Examples include computation of brain deformation during cranial image-guided surgery [3]–[5], correction of artifact due to patient motion during contrast-enhanced subtraction imaging [6]–[8], kinematic modeling of abdominal organ motion during respiration [9], and correction of geometric distortion in MR images [10]. Nonrigid methods are also useful for matching of images from different subjects [11] and generating an average atlas [12], [13].

One major problem with nonrigid image registration techniques is their high computational cost. Because of this, these methods have found limited application to clinical situations where fast execution is required, e.g., intraoperative imaging. This is disappointing since some of the most interesting and important problems involve intraoperative imaging, e.g., computation of brain deformation during cranial image-guided surgery. Other groups have already published results of high-performance computing hardware to similar problems, including rigid registration on a cluster of symmetric multiprocessors [14], nonrigid registration on a massively parallel architecture [15], and nonrigid registration and segmentation on a multiprocessor workstation [16]. In this paper, we address the problem of high computation cost using a parallel implementation that takes full advantage of modern large-scale shared-memory multiprocessor computer architectures. We describe in detail the strategies employed to parallelize an algorithm based on free-form deformations using B-spline interpolation [8] and demonstrate how the execution time of a nonrigid image registration algorithm can be dramatically reduced. To illustrate the benefits of parallelization, we perform an experimental analysis of its scaling properties for three biomedical applications. We do not consider in this work any of the other successful nonrigid image registration algorithms, such as methods based on fluid transformations [17], elastic models [18], or optical flow [19]. For surveys of the field, the interested reader is refered to [20]–[22].

II. Shared-Memory Parallel Programming

All CPUs in a shared-memory multiprocessor computer share the same main memory and, thus, can work on the same data concurrently. This is an important advantage of this type of hardware over a cluster of independent workstations, as it largely eliminates the need for explicit message passing between concurrent tasks. It is this property in particular that makes it com-

Manuscript received December 20, 2001; revised January 29, 2002. The work of T. Rohlfing was supported by the National Science Foundation under Grant EIA-0104114. This work was supported by CBYON, Inc., Mountain View, CA.

The authors are with the Image Guidance Laboratories, Department of Neurosurgery, Stanford University, Stanford, CA 94305-5327 USA (e-mail: rohlfing@stanford.edu; calvin.maurer@igl.stanford.edu).

Digital Object Identifier 10.1109/TITB.2003.808506

paratively easy to parallelize an originally sequential piece of software on a shared-memory system, especially when compared to other parallel system architectures, e.g., clusters of independent workstations programmed using Parallel Virtual Machine (PVM) [23].

A programming paradigm tailored to shared-memory multiprocessor computers is *multithreaded programming* [24] where each application can branch into independent, potentially concurrent threads. Programming libraries and operating system support for multithreaded programming are available today on most platforms, including virtually all available Unix variants (including Mac OS X) and recent versions of Windows. The precise differences between multithreaded programming and process-based *multitasking* are beyond the scope of this paper. Suffice it to say that, by default, all threads in a multithreaded application share the same address space[1] and that creating a certain number of threads causes substantially less overhead than creation of the same number of processes (we found a factor of 50 for the system we used in the present study). Still, it is worth noting that there is a certain amount of overhead associated with the handling of multiple threads, so the performance gain achieved by parallelization must outweigh this overhead in order to be useful.

Multithreaded applications profit from shared-memory multiprocessors as all threads can run concurrently on the available CPUs. In case one thread gets blocked, for example, in order to achieve exclusive access to shared data, then another thread of the application can execute in its place. We will show how an intensity-based nonrigid image registration algorithm can be broken into parallel tasks that run as (almost) independent threads. The immediate benefit of this is that these threads can run concurrently on different CPUs, thus reducing the real-world time the user has to wait for the result of the registration algorithm. For the purpose of this paper, it is, therefore, practical to understand each thread as a "virtual CPU" that can perform computations at the same time as other threads do other computations.

III. Intensity-Based Nonrigid Image Registration

Intensity-based image registration algorithms optimize a function that quantifies the degree of mutual similarity between two images. The parameters of the similarity function are the degrees of freedom of a transformation that maps the coordinate space of one image (the floating image) into the space of a second image (the reference image). Two aspects that make nonrigid image registration a very time consuming task are the typically large sizes of the images, resulting in high computational cost of evaluating the similarity measure, and the sometimes large number of degrees of freedom, resulting in frequent repetition of the evaluation. We describe below an intensity-based nonrigid registration algorithm, and discuss how both of these computational issues can be addressed using the techniques of multithreaded programming.

[1]In more detail, all threads share the same text and heap memory (program code and global data), while each thread has its own private segment of the stack.

A. Image Similarity

Our algorithm computes an information-theoretic similarity measure, normalized mutual information (NMI) [25], using discrete, integer-valued bins in a two-dimensional (2-D) joint histogram [26], [27]. For each voxel in the reference image, the corresponding voxel in the floating image is determined under the current coordinate transformation. The intensity of the reference voxel in each pair determines the horizontal location of that pair in the joint histogram while the intensity of the interpolated floating voxel determines the vertical position of the pair. The histogram bin indexed by the voxel intensities is incremented by one. After all voxel pairs have been added, the approximate joint and marginal probability distributions are computed from the distribution of voxel pairs in the histogram bins. Finally the marginal and joint entropies and NMI are computed from the probability distributions.

An obvious but important observation is that two or more joint histograms, computed from nonoverlapping partitions of the reference image data, can be combined into a single histogram for the complete image by adding the values in corresponding bins. This requires identical numbers of bins and identical value ranges among all the partial histograms, which can easily be achieved.

B. Coordinate Transformation

Our implementation of nonrigid registration is a modified version [6], [9] of an algorithm first described by Rueckert *et al.* [8]. It also incorporates some features of methods presented by other groups [10]. The geometric transformation model is based on free-form deformations [28] represented by multilevel B-splines [29], defined on a uniform three-dimensional (3-D) control point grid (CPG). The control points $\phi_{i,j,k}$ are moved independently and define a continuous deformation of the coordinate space by interpolation between them using 3-D third-order B-splines. Specifically, the transformed coordinate of a location (x, y, z) is computed as

$$\mathbf{T}(x,y,z) = \sum_{l,m,n=0}^{3} B_l(u) B_m(v) B_n(w) \phi_{i+l,j+m,k+n}. \quad (1)$$

Here, i, j, and k denote the indexes of the control point cell containing (x, y, z), and u, v, and w are the relative positions of (x, y, z) inside that cell in the three dimensions. These are defined as

$$i = \left\lfloor \frac{x}{\delta_x} \right\rfloor - 1,\ j = \left\lfloor \frac{y}{\delta_y} \right\rfloor - 1,\ k = \left\lfloor \frac{z}{\delta_z} \right\rfloor - 1 \quad (2)$$

and

$$u = \frac{x}{\delta_x} - (i+1),\ v = \frac{y}{\delta_y} - (j+1),\ w = \frac{z}{\delta_z} - (k+1) \quad (3)$$

where δ_x, δ_y, and δ_z are the distances between the control points in the three dimensions. The functions B_0 through B_3 are the approximating third-order spline polynomials [29].

To ensure efficient computation of the B-spline transformation, we exploit the fact that the axes of the CPG are parallel to

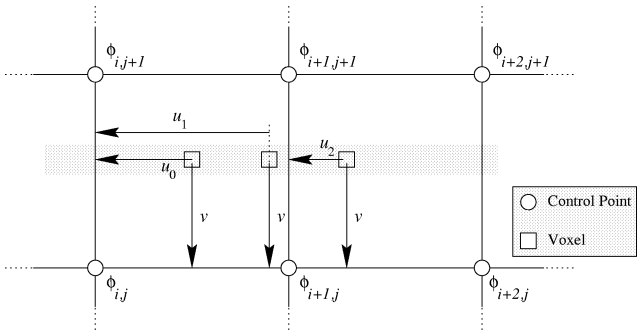

Fig. 1. Precomputation of CPG coefficients in 2-D. The CPG cell index (i, j) is constant for all voxels within the same cell; only i increases by one when crossing a cell boundary. For all voxels in the same row (*horizontal gray bar*), the relative offset u of each voxel in x direction is constant and can be precomputed. The relative offset v of the voxel row in y direction is constant and identical for all voxels in that row and can be precomputed. The 3-D case is a straightforward extension of the situation shown here.

the axes of the reference image. The coefficients i, u, and $B_0(u)$ through $B_3(u)$ for all columns of the reference image can, therefore, be precomputed. Similarly, j, v, and $B_0(v)$ through $B_3(v)$ are precomputed for all rows of the reference image and k, w, and $B_0(w)$ through $B_3(w)$ for all planes. Fig. 1 illustrates the underlying principles in the simplified 2-D case.

Using the precomputed coefficients so far, we can further precompute parts of (1). For all voxels of the same image row inside the same cell of the CPG, the coefficients i, j, and k, as well as v and w, are constant, while only u varies from voxel to voxel. Therefore, $B_m(v)$ and $B_n(w)$, as well as their product with the components of the control point positions $\phi_{i+l,j+m,k+n}$, are constant and need only be computed once per CPG cell. Equation (1) can thus be rewritten as

$$\mathbf{T}(x,y,z) = \sum_{l=0}^{3} B_l(u)\hat{\phi}_{i+l} \tag{4}$$

where

$$\hat{\phi}_{i+l} = \sum_{m=0}^{3}\sum_{n=0}^{3} B_m(v)B_n(w)\phi_{i+l,j+m,k+n} \tag{5}$$

is identical for all voxels in one row that are located within the same control point cell. Furthermore, when entering the next CPG cell in the x-direction, i.e., when incrementing i by one, three out of four precomputed vectors $\hat{\phi}$ can be reused after a simple index transformation. The easiest way to achieve this is to precompute the sequence of all vectors $\hat{\phi}$ required for the current row and move along this sequence as the row traverses the CPG cells.

Computing the free-form deformation for all voxels in the reference image is the dominant computational task in our algorithm. Therefore, although the implementation of the above optimizations is not necessarily trivial, their impact is well worth the effort. Our experience is that aggressive precomputation of elements of the B-spline transformations reduces computation time by approximately 50%.

C. Multilevel Deformation With Adaptive Grid Refinement

Before parallelizing an algorithm, an essential first step is to improve its efficiency as much as possible. Aggressive precomputation of elements of the B-spline transformation is one way to improve efficiency, as discussed in Section III-B above. Another important strategy is to use a multiresolution deformation approach with adaptive grid refinement [6]. Registration starts with a rather coarse CPG, typically 40 mm for clinical data. The relatively small number of parameters enables rapid computation, while the large spacing of the grid allows capture of large scale deformations of the anatomy. As registration proceeds, the grid is successively refined by reducing the space between the control points by factors of two, thus modeling increasingly localized deformation. Refinement itself is done in a way that preserves precisely the deformation generated by the coarse grid before refinement. The formulas to achieve this are based on some fairly simple substitutions that can be found in [9], generalized from the 2-D case described by Lee *et al.* [29].

In order to keep computation times low even with large numbers of parameters required by fine-resolution control point grids, those control points that do not affect any regions of interest are fixed. Their locations are then no longer degrees of freedom of the transformation and need not be considered during registration. In particular, a control point is fixed if the $4 \times 4 \times 4$ control point neighborhood it influences has little structure (intensity variation) as determined using a local entropy criterion described in [6]. This efficiently fixes control points in the image background. In practice, the adaptive disabling of control points typically leads to a reduction of the number of degrees of freedom of the nonrigid transformation by a factor of at least two, sometimes as much as five, depending on the particular images. Not only does this reduce the number of parameters to be considered during gradient estimation considerably, but it also reduces the overall dimension of the search space, thus on average reducing the number of optimization steps required to achieve convergence.

D. Optimization

The degrees of freedom of a B-spline based transformation are the coordinates of the control points $\phi_{i,j,k}$. They are interpreted as absolute positions and initialized as

$$\phi^{(0)}_{i,j,k} := (i\delta_x, j\delta_y, k\delta_z). \tag{6}$$

For a CPG with $N_x \times N_y \times N_z$ control points, the total number of parameters is $N = 3N_xN_yN_z$. For finding the optimal parameters that maximize the NMI image similarity measure between reference and floating image, we employ an iterative process using a multiresolution line search algorithm [9]. Our method is a variant of the Downhill–Simplex algorithm [30], [31] restricted to the direction of the steepest ascent and consists of the following steps.

1) Compute global image similarity.
2) Estimate gradient of image similarity.
3) Search for optimum along gradient direction using repeated evaluation of global image similarity.
4) Repeat from step 1) until no further improvement can be achieved.

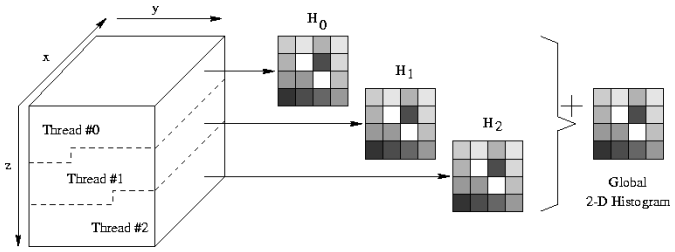

Fig. 2. Partitioning of 3-D image and assignment of partitions to threads for global image similarity computation. Each thread generates a 2-D joint histogram from its partition of the reference image and the corresponding region of the floating image. These partial histograms are added to form the global joint histogram for NMI computation.

The key steps of the above procedure involve computing the similarity measure between reference and floating image as well as an estimate of its gradient with respect to the transformation parameters. These are the two steps accounting for the bulk of the algorithm's computational cost, although for very different reasons. Computation of image similarity involves processing large amounts of data, in particular evaluating the nonrigid coordinate transformation for every single voxel in the reference image. On the other hand, the gradient computation step deals with relatively small chunks of data only and thus has to evaluate the nonrigid transformation only for a small number of voxels at a time. However, the number of these chunks is equal to the number of parameters of the coordinate transformation, which is typically very large. So while evaluation of the global image similarity measure is a single expensive task, gradient computation is a repetition of many tasks, each single one of which is rather inexpensive. It is for this difference that two different strategies are applied in order to parallelize the two computations.

IV. Parallel Implementation

A. Computation of Image Similarity

Evaluation of the similarity measure is a global operation that involves the complete image. We parallelize this step by breaking the data into equally sized partitions. Each thread is assigned one of these partitions and computes its contribution to the similarity measure. Fig. 2 illustrates the principle. In detail, we assign to each thread a continuous range of *rows* of the reference image. This allows us to efficiently generate transformed grid coordinates for a complete image row at a time by exploiting constant terms in the B-spline function, as described in (4).

The voxel pairs encountered by each thread are stored in separate 2-D histograms (Fig. 2). After finishing its part of the computation, each thread adds the entries of the histogram it created to the global histogram. This step needs to be kept mutually exclusive and is, therefore, protected by a mutex lock. It makes sense, however, to have each thread add its data to the global data structure rather than having the main thread collect all partial results. This way, threads that finish before the others can use the extra time for completing part of a task that, otherwise, the main thread would have to perform sequentially.

B. Gradient Computation

Estimation of the gradient Δf of the image similarity measure is achieved by means of the common finite-difference approximation

$$(\Delta f)_i = \frac{f(\mathbf{x}+\boldsymbol{\delta}_i) - f(\mathbf{x}-\boldsymbol{\delta}_i)}{2\delta} \tag{7}$$

where $\boldsymbol{\delta}_i$ is the vector that has δ as the ith element and all zeroes otherwise. Due to the compact support of the B-spline functions (moving any control point affects only its $4 \times 4 \times 4$ neighborhood), the computation of $f(\mathbf{x}+\boldsymbol{\delta}_i)$ and $f(\mathbf{x}-\boldsymbol{\delta}_i)$ is identical to the computation of $f(\mathbf{x})$ outside that neighborhood of the control point controlled by the ith parameter. One can, therefore, precompute the 2-D histogram corresponding to $f(\mathbf{x})$ and substitute only the voxel pairs that are affected by moving the current control point.

Hence, computing any particular element of the gradient Δf is a local operation that needs to consider only a small fraction of the image data per parameter. It therefore makes no sense to assign parts of the image data to multiple threads as the computational cost of gradient computation is caused by the large number of parameters (up to several hundreds of thousands frequently occur in practically relevant cases). This step is instead parallelized by assigning an equal number of the parameters to each of the threads which then compute the respective components of the gradient. We do not start a new thread for each parameter as this would cause substantial computational overhead. For the same reason, and because the algorithm would not scale for more than two CPUs, we do not let one thread compute the $+\delta$ contribution while another thread computes the $-\delta$ part. Instead, we create the given number of threads and have each work on a subset of parameters.

One of the advanced features of our algorithm is the adaptive fixing of control points in areas with little image information in order to reduce serial processing time (see Section III-C). As an immediate consequence, the sequence of fixed and moving control points (and therefore parameters) depends on the particular image. It can, therefore, not be assumed that equally sized continuous sets of parameters all contain the same number of active parameters. In order to distribute those evenly among all threads, each runs through a list of all parameters, counting the ones that are variable. Out of these, it chooses the ones that have an index i congruent to the respective threads index t when divided by the number of threads T, i.e., $t \bmod T = i$. This principle is illustrated for three threads in Fig. 3.

It is obvious that a thread with small index is more likely to encounter more parameters to deal with than a thread with a higher index. We therefore start threads in the order of their index and collect their results in the reverse order (see Fig. 4).

C. Thread Implementation and Run Time Analysis

We implement and evaluate the algorithm described above on an SGI Origin 3800 computer with 128 MIPS R12K processors running at 400 MHz (Silicon Graphics, Mountain View, CA). The operating system is Irix 6.5 with a single kernel image architecture. Our algorithm is coded entirely in C++ and compiled using version 7.3.1.2 of the MIPSpro compiler suite. Threads are implemented using the POSIX threads library. POSIX

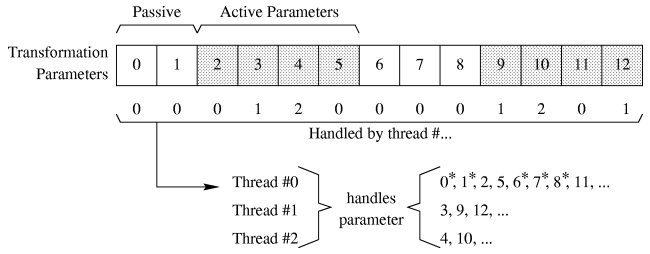

Fig. 3. Assignment of function parameters to threads for gradient computation (three threads). All passive parameters are handled by thread #0, i.e., the respective gradient vector elements are set to zero. All active parameters are assigned to the threads using a cyclic schedule. For the passive parameters marked with a * for thread #0, the respective gradient component is set to zero, consuming almost no computation time.

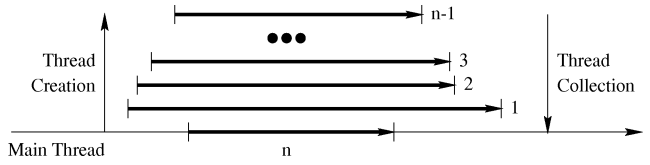

Fig. 4. Timing of thread creation and completion. Threads that are expected to run longer than others are started first and their results collected last. The subtask with the highest index n is not actually started as a thread but runs as part of the main thread.

threads were chosen over the proprietary $\mathrm{sproc}()$ interface provided by SGI in order to ensure portability. For example, we also successfully use the identical software package compiled with version 3.0.2 of the GNU C++ compiler on multiprocessor Sun workstations (Sun Microsystems, Palo Alto, CA) running Solaris 7. Also, the $\mathrm{sproc}()$ interface with kernel-based threads is commonly recommended for I/O-intensive applications while the user-space POSIX threads are recommended for CPU-intensive applications, due to the substantially reduced overhead of the latter, which further supports our decision to use POSIX threads rather than the $\mathrm{sproc}()$ interface.

The real time $T(n)$ required to complete a task on n parallel processors can be approximated as

$$T(n) = A + \frac{B}{n} \tag{8}$$

where A is the serial (nonparallelizable) portion of the computation and B is the parallel portion. The parameters A, and B can be determined by linear regression of measured CPU times versus the inverse number of CPUs $1/n$. Conversely, one can define the speedup c_n as one out of several metrics for parallel performance. Given the above definitions, the speedup for n CPUs can be expressed as

$$c_n = \frac{T(1)}{T(n)} = \frac{A+B}{A+\frac{B}{n}}. \tag{9}$$

This relationship, usually expressed as an inequality to account for parallelization overhead, is generally known as "Amdahl's Law" [32]. It is obvious from this expression that the speedup of a parallel algorithm does not continue to increase with increasing number of processors. Since only the parallel component scales while the time required to complete the serial component remains constant, there is a theoretical limit for the maximum parallel speedup, denoted as

$$c_\infty = \lim_{n\to\infty} c_n = \frac{A+B}{A} = 1 + \frac{B}{A}. \tag{10}$$

A technical problem for the performance analysis of a parallel algorithm arises from the requirement to accurately measure computation time. Although one can easily determine the total CPU time of the registration process with a call to the $\mathrm{times}()$ library function, this is not the information we are after. In fact, the returned value represents the *sum* of the processing times spent by *all processors*, which is virtually invariant with respect to the number of CPUs. Instead, we need to determine the computation time of the *longest running thread*. For obvious reasons, one cannot simply use a stop watch for this, as its measurements would be strongly affected by the system load during execution of the registration task.

Operating system support for computation time accounting on the other hand is only available for processes and not for user-space threads. Fortunately, using the aforementioned $\mathrm{sproc}()$ function, concurrent child processes can be created that share the address space of the parent process and from a programmer's point of view behave very much like threads. However, as each $\mathrm{sproc}()$ process is a separate kernel entity, computation time accounting can be performed by comparing the computation time of the parent process versus the sum of computation times of parent and all child processes. Using the $\mathrm{sproc}()$ interface, the $\mathrm{times}()$ function from the standard C library provides convenient access to all necessary information. Thus implemented are two versions of our algorithm: one for clinical application using the portable and more efficient POSIX threads and one for performance measurement using the less efficient and proprietary but accountable $\mathrm{sproc}()$ interface.

V. Application Results

To empirically investigate the scaling properties of the nonrigid image registration algorithm described in this paper, it was applied to image data from three biomedical applications. For each application, four registration problems were randomly selected from the available data. Each registration was performed several times with varying degrees of parallelism: The algorithm was run serially and with two- through 16-, 32-, 48-, and 64-fold parallelism.[2] Due to the the operating system configuration on the supercomputer used in this study, parallel computation on more than 64 CPUs was not available to us. Execution times were obtained with the implementation using the $\mathrm{sproc}()$ function for thread creation.

A. Intraoperative Brain Deformation Analysis

Purpose: All image-guided neurosurgical systems that use preoperative images assume that the head and its contents behave as a rigid body, i.e., that the intraoperative positions of anatomical structures of interest are related to the positions of these structures in the preoperative images by a rigid-body trans-

[2]It should be pointed out that the creation of threads has a certain computational overhead. Furthermore, no more threads than the available number of CPUs can run concurrently. Therefore, the number of threads created should in general not exceed the number of CPUs.

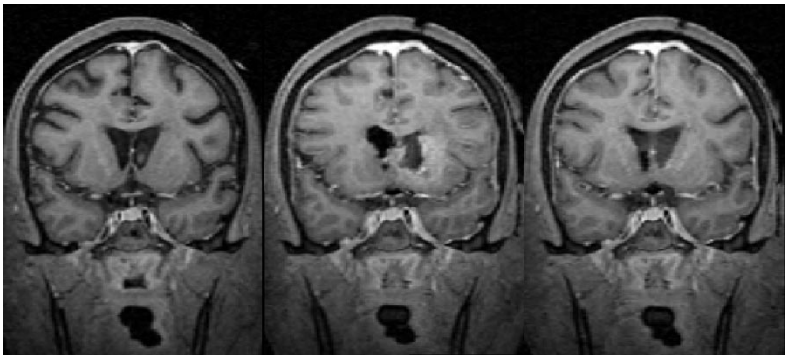

Fig. 5. Pre- and intraoperative brain MR images. Left: preoperative image; center: intraoperative image after rigid registration; right: intraoperative image after nonrigid registration.

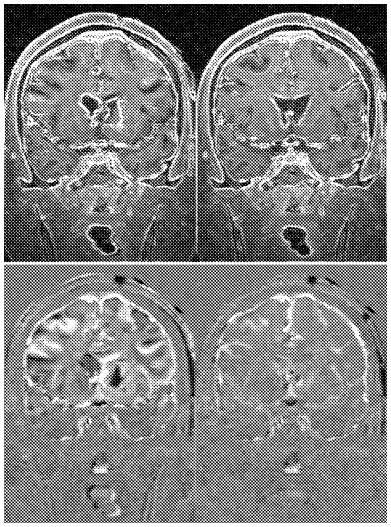

Fig. 6. Combined visualization of pre- and intraoperative brain MRI. Top row: edges extracted from intraoperative image overlaid onto corresponding preoperative image; bottom row: subtraction of corresponding pre- and intraoperative image; left column: after rigid registration; right column: after nonrigid registration. Note that after nonrigid registration, the sign of almost all nonzero pixels in the subtraction image is positive. This is the result of contrast applied during surgery; it does not indicate misalignment of pre- and intraoperative image.

formation (sometimes scaling is also used to account for incorrect image voxel dimensions). Brain deformation between the time of imaging and the time of surgery, or during surgery, invalidates this assumption and consequently introduces an important source of error. Image-guided neurosurgical procedures can now be performed using intraoperative MR imaging [33], [34]. Brain deformation is also a concern for such procedures because it may be desirable to make intraoperative use of preoperative images (e.g., from other modalities) or information derived from preoperative images (e.g., a surgical plan). It is obviously important to compute the deformation transformation quickly so that surgery is not delayed by waiting for the registration algorithm to execute.

Image Data: Fig. 5 shows coronal slices from pre- and intraoperative images of a patient previously used for quantitative analysis of brain deformation under craniotomy [3], [5]. This is one out of four patients for whom we performed parallelized nonrigid registrations as part of this paper.

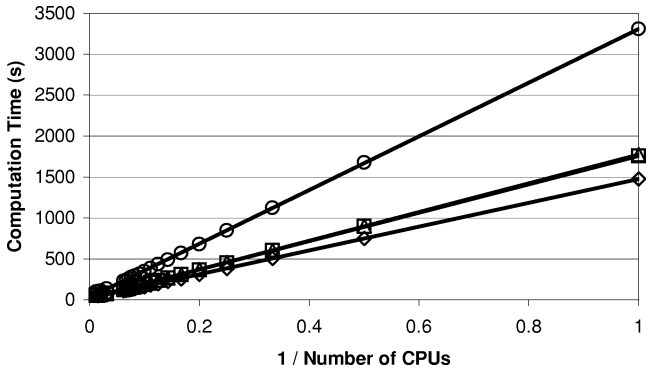

Fig. 7. Computation time versus inverse number of CPUs for nonrigid registration of pre- and intraoperative 3-D MR head images. Each symbol represents one subject; the bold lines represent the result of the linear regression fit of the measured data per subject.

TABLE I
COEFFICIENTS OF PARALLELIZATION FOR NONRIGID REGISTRATION OF PRE-AND INTRAOPERATIVE 3-D MR HEAD IMAGES. REGRESSION COEFFICIENT WAS $R > 0.9995$ FOR ALL FOUR CASES. UNITS OF $T(\cdot)$, A, AND B ARE TIME IN SECONDS

Patient	$T(1)$	$T(4)$	$T(16)$	$T(64)$	A (serial)	B (parallel)	c_∞
1	1477	381	107	50	17	1459	87
2	1759	450	125	60	19	1740	93
3	1776	458	125	55	18	1758	99
4	3311	845	227	102	25	3286	132
Mean	2081	533	146	67	20	2061	104
Std. Dev.	832	210	55	24	3.2	717	

Results: All nonrigid brain registrations were performed starting with an initial CPG spacing of 30 mm that was refined to 15 mm. The original image data was resampled to 2 mm voxel size at the first deformation level and 1 mm voxel size at the second level. Visual inspection of edge overlay and subtraction images such as those illustrated in Fig. 6 supports that the resulting deformation transformations are relatively accurate. Using 64 CPUs, computing the nonrigid coordinate transformation between the pre- and intraoperative images shown in Fig. 6 took about 60 seconds. For the three other pre- to intraoperative brain registrations we performed, the computations took between 50 and 102 s (mean 67 s). The theoretical lower bound for the computation time of all four cases was below 30 s; the theoretical upper limit for the parallel

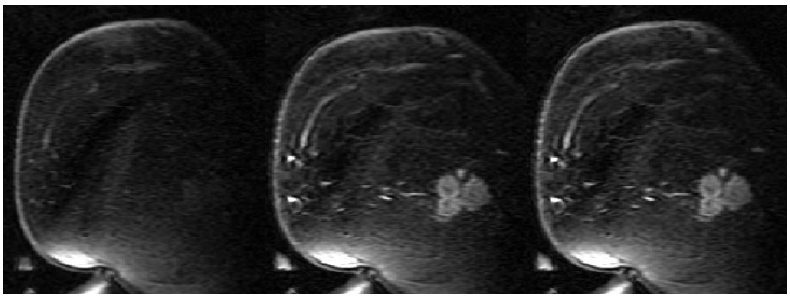

Fig. 8. Pre- and postcontrast breast MR images. Left: precontrast image; center: postcontrast image after rigid registration; right: postcontrast image after nonrigid registration. A massive contrast-enhancing lesion can easily be detected when comparing pre- and postcontrast images.

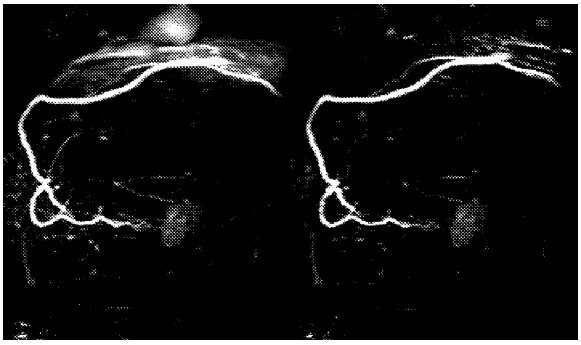

Fig. 9. Maximum-intensity projection of MR mammography subtraction image. Left: after rigid registration; right: after nonrigid registration.

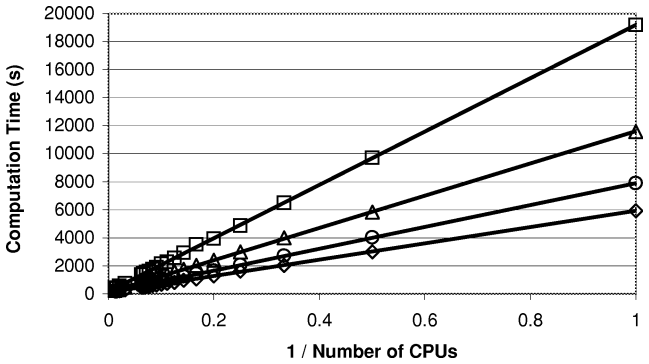

Fig. 10. Computation time versus inverse number of CPUs for MR mammography registration. Each symbol represents one subject; the bold lines represent the result of the linear regression fit of the measured data per subject.

speedup was between 87 and 132 (mean 104). Fig. 7 shows a plot of the computation times versus the inverse number of CPUs with a linear regression fit. Table I gives all computation times and the performance coefficients determined using linear regression.

B. Contrast-Enhanced MR Mammography

Purpose: By subtracting images of the same patient acquired before and after the injection of contrast, vascular structures and contrast-enhancing lesions can be localized and visualized. A frequent problem with such techniques is artifacts in the subtraction image caused by patient motion between the pre- and postcontrast scans. Since this motion typically involves local deformations, especially in areas such as the abdomen or the breast, nonrigid registration is required to correct for patient motion [6]–[8].

Image Data: As part of the present investigation representing such problems, we have included four patients from a previous study on artifact reduction in MR mammography [6]. Fig. 8 shows typical sagittal slices from one of these patients.

Results: The CPG was refined to a final resolution of 5 mm, and the original image data was used during the last step of the registration process. When the subtraction image is rendered in 3-D using maximum intensity projection (MIP), substantial motion artifacts become apparent that cannot be compensated for by rigid registration (Fig. 9, left image). After nonrigid registration, these artifacts disappear almost completely (Fig. 9, right image) or are, in more complicated cases, at least substantially reduced. Absolute computation times are appreciably higher than for the intraoperative brain image registration (Section V-A), because the CPG was refined to a finer resolution. Fig. 10 shows a plot of the computation times versus the inverse number of CPUs with a linear regression fit. The parameters of the fit are given in Table II. The theoretical lower bound for the

TABLE II
COEFFICIENTS OF PARALLELIZATION FOR NONRIGID REGISTRATION OF PRE- AND POSTCONTRAST BREAST MR IMAGES. REGRESSION COEFFICIENT WAS $R > 0.9995$ FOR ALL FOUR CASES. UNITS OF $T(\cdot)$, A, AND B ARE TIME IN SECONDS

Patient	$T(1)$	$T(4)$	$T(16)$	$T(64)$	A (serial)	B (parallel)	c_∞
1	5938	1609	477	193	102	5838	58
2	19195	4875	1363	436	183	19014	105
3	11588	2990	819	288	108	11488	107
4	7878	2061	581	221	106	7799	75
Mean	11150	2884	810	284	125	11034	89
Std. Dev.	5853	1447	396	109	34	5034	

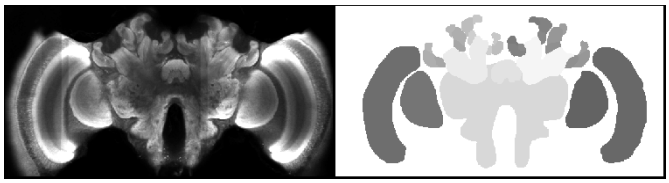

Fig. 11. Central axial slice from individual confocal microscopy image (left) of a bee brain and corresponding segmented label image (right). Every gray level in the label image represents a different anatomical structure. Due to limitations of reproduction, different gray levels may look alike.

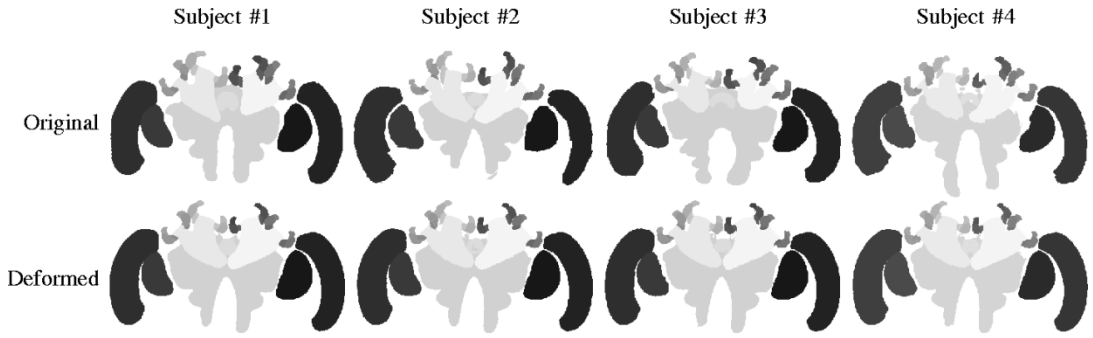

Fig. 12. Original and deformed bee brain label images. Top row: central axial slices from randomly selected original label images; bottom row: corresponding slice from the final image after nonrigid registration to a common reference.

computation time of all four cases was 183 s or less[3]; the theoretical upper limit for the parallel speedup was between 58 and 107 (mean 89).

C. Intersubject Brain Atlas Generation

Purpose: Images from a population of independent subjects can be compiled into an average image that represents not only average voxel values, but also average object shapes, using an iterative nonrigid registration process [13]. Such average images can for example be used to quantify interindividual shape variation or to compare different subgroups of individuals.

Image Data: For this study, confocal microscopy and segmented (label) images of 20 brains from adult, foraging honeybees served as subjects. Details on the imaging process can be found in [13]. The final image volume contained 84–114 slices (sections) with a thickness of 8 μm. Each slice had 610–749 pixels in the x direction and 379–496 pixels in the y direction with pixel size 3.8 μm. Subsequently, 22 areas of interest were traced manually on each slice. Examples of confocal microscopy and label images are shown in Fig. 11. The label images were used for registration rather than the original microscopy images because the latter suffer from severe intensity artifacts that complicate intensity-based nonrigid registration.

Results: All individual label images were registered nonrigidly to a common reference, the final average shape atlas. For the present study, four individuals were randomly selected and registered using our parallel algorithm. Examples of characteristic slices from these brains before and after registration are shown in Fig. 12. The dependence of registration times on the number of CPUs is shown in Fig. 13 for the first nonrigid registration iteration applied to four randomly selected bee brains. This particular iteration used a CPG spacing of 120 μm and image data resampled to 16 μm voxel size. The parallel scaling coefficients derived from fitting the computation times

[3]Rueckert *et al.* [8] reported between 15 and 30 min run time on a single-CPU Sun Ultra10 workstation for their original implementation of the nonrigid registration method used in this paper. The mean execution time of the four patients in our study using one CPU is approximately 185 min. However, their patient data had 30–40 slices compared to 60 slices in our data. Also, their image matrix was 256 × 256 (compared to 512 × 512), and they computed deformations on a 10 mm CPG (compared to 5 mm). Given the substantially larger images in our study (six times more voxels) and the smaller CPG spacing that we used (two times higher resolution), the performance reported by Rueckert and the data presented in this paper are consistent.

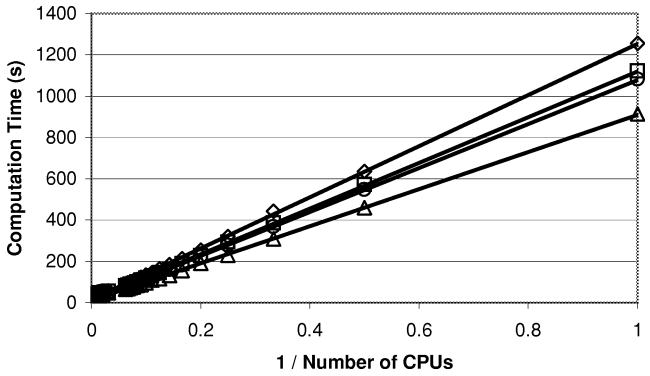

Fig. 13. Computation time versus inverse number of CPUs for bee brain registration. Each symbol represents one subject; the bold lines represent the result of the linear regression fit of the measured data per subject.

TABLE III
COEFFICIENTS OF PARALLELIZATION FOR BEE BRAIN ATLAS REGISTRATION. REGRESSION COEFFICIENT WAS $R > 0.9995$ FOR ALL FOUR CASES. UNITS OF $T(\cdot)$, A, AND B ARE TIME IN SECONDS

Bee	$T(1)$	$T(4)$	$T(16)$	$T(64)$	A (serial)	B (parallel)	c_∞
1	1255	319	89	47	12	1243	105
2	1121	293	81	49	12	1109	93
3	914	231	66	36	10	900	91
4	1082	277	81	53	12	1066	90
Mean	1093	280	79	46	12	1079	91
Std. Dev.	140	37	10	7.2	0.9	122	

are given in Table III. The theoretical lower bound for the computation time of all four cases was 12 s or less; the theoretical upper limit for the parallel speedup was between 90 and 105 (mean 91).

VI. CONCLUSION

The work presented in this paper addresses one of the major problems of clinical application of nonrigid image registration, that is, its high computational cost. By using a currently available shared-memory multiprocessor computer, we reduced execution times from hours to less than 1 min for several applications.

Concerning the practical usefulness of our approach, clearly there will not be shared-memory computers with large numbers of CPUs (64 or more) in operating rooms anytime soon. Nevertheless, given the rapidly increasing availability of high-speed

networks with guaranteed quality of service, e.g., the Internet2, the computational power of supercomputers is becoming available for clinical applications, e.g., computation of brain deformation during cranial image-guided surgery. Also, it is important to note that while most supercomputers installed today are already heavily utilized, a nonrigid image registration task such as one of the applications reported in this paper requires only a relatively small amount of total time to finish on one of these machines. It is usually possible to give such jobs a high priority, resulting in virtually exclusive access to the CPU resources for a short time. One can, therefore, easily conceive a scenario where supercomputing centers perform intraoperative image registrations for remote clinical sites in near-real time on demand. In between these high-priority but low-duration tasks, their resources would remain available to other scientific computations that take substantially more time to compute but for which waiting for the result for a few more minutes can be easily tolerated.

ACKNOWLEDGMENT

The authors thank B. Lewis for his generous help and expert advice on thread programming and the reviewers of this paper for their valuable comments and helpful suggestions. Computations were performed on an SGI Origin 3800 in the Stanford University Bio-X core facility for Biomedical Computation. Breast MR images were provided by M. Jacobs and D. Bluemke, Department of Radiology, The Johns Hopkins University, Baltimore, MD. Human brain MR images were provided by W. Hall, Department of Neurosurgery, and C. Truwit, Department of Radiology, University of Minnesota, Minneapolis, MN, and D. Hill and T. Hartkens, Division of Radiological Sciences, Guy's Hospital, King's College London, London, U.K. Bee brain images were provided by R. Brandt and R. Menzel, Institute for Neurobiology, Free University of Berlin, Berlin, Germany.

REFERENCES

[1] J. M. Fitzpatrick, D. L. G. Hill, and C. R. Maurer Jr., "Image registration," in *Handbook of Medical Imaging, Volume 2: Medical Image Processing and Analysis*, M. Sonka and J. M. Fitzpatrick, Eds. Bellingham, WA: SPIE, 2000, ch. 8, pp. 447–513.

[2] J. B. West, J. M. Fitzpatrick, M. Y. Wang, B. M. Dawant, C. R. Maurer Jr., R. M. Kessler, R. J. Maciunas, C. Barillot, D. Lemoine, A. Collignon, F. Maes, P. Suetens, D. Vandermeulen, P. A. van den Elsen, S. Napel, T. S. Sumanaweera, B. Harkness, P. F. Hemler, D. L. G. Hill, D. J. Hawkes, C. Studholme, J. B. A. Maintz, M. A. Viergever, G. Malandain, X. Pennec, M. E. Noz, G. Q. Maguire Jr., M. Pollack, C. A. Pelizzari, R. A. Robb, D. Hanson, and R. P. Woods, "Comparison and evaluation of retrospective intermodality brain image registration techniques," *J. Comput. Assist. Tomography*, vol. 21, no. 4, pp. 554–566, 1997.

[3] T. Hartkens, D. L. G. Hill, C. R. Maurer Jr., A. J. Martin, W. A. Hall, D. J. Hawkes, D. Rueckert, and C. L. Truwit, "Quantifying the intraoperative brain deformation using interventional MR imaging," in *Proc. Int. Soc. Magnetic Resonance Medicine*, vol. 8, 2000, p. 51.

[4] D. L. G. Hill, C. R. Maurer Jr., A. J. Martin, S. Sabanathan, W. A. Hall, D. J. Hawkes, D. Rueckert, and C. L. Truwit, "Assessment of intraoperative brain deformation using interventional MR imaging," in *Proc. Medical Image Comput. Comput.-Assisted Intervention*, C. J. Taylor and A. C. F. Colchester, Eds. Berlin, Germany, 1999, pp. 910–919.

[5] C. R. Maurer Jr., D. L. G. Hill, A. J. Martin, H. Liu, M. McCue, D. Rueckert, D. Lloret, W. A. Hall, R. E. Maxwell, D. J. Hawkes, and C. L. Truwit, "Investigation of intraoperative brain deformation using a 1.5 Tesla interventional MR system: Preliminary results," *IEEE Tran. Med. Imag.*, vol. 17, pp. 817–825, Oct. 1998.

[6] T. Rohlfing and C. R. Maurer Jr., "Intensity-based nonrigid registration using adaptive multilevel free-form deformation with an incompressibility constraint," in *Proc. 4th Int. Conf. Medical Image Comput. Comput.-Assisted Intervention*, vol. 2208, Lecture Notes in Computer Science, W. Niessen and M. A. Viergever, Eds. Berlin, Germany, 2001, pp. 111–119.

[7] T. Rohlfing, C. R. Maurer Jr., and J. Beier, "Correction of motion artifacts in three-dimensional CT-DSA using constrained adaptive multilevel free-form registration," in *Proc. Computer Assisted Radiology and Surgery*, Excerpta Medica, International Congress Series 1230, H. U. Lemke, M. W. Vannier, K. Inamura, A. G. Farman, and K. Doi, Eds., Amsterdam, The Netherlands, 2001, pp. 350–355.

[8] D. Rueckert, L. I. Sonoda, C. Hayes, D. L. G. Hill, M. O. Leach, and D. J. Hawkes, "Nonrigid registration using free-form deformations: Application to breast MR images," *IEEE Trans. Med. Imag.*, vol. 18, pp. 712–721, Aug. 1999.

[9] T. Rohlfing, C. R. Maurer Jr., W. G. O'Dell, and J. Zhong, "Modeling liver motion and deformation during the respiratory cycle using intensity-based free-form registration of gated MR images," in *Proc. SPIE Medical Imaging: Visualization, Display, Image-Guided Procedures*, vol. 4319, , S. K. Mun, Ed., 2001, pp. 337–348.

[10] C. Studholme, R. T. Constable, and J. S. Duncan, "Accurate alignment of functional EPI data to anatomical MRI using a physics-based distortion model," *IEEE Trans. Med. Imag.*, vol. 19, pp. 1115–1127, Nov. 2000.

[11] A. Guimond, A. Roche, N. Ayache, and J. Meunier, "Three-dimensional multimodal brain warping using the demons algorithm and adaptive intensity corrections," *IEEE Trans. Med. Imag.*, vol. 20, pp. 58–69, Jan. 2001.

[12] A. Guimond, J. Meunier, and J.-P. Thirion, "Average brain models: A convergence study," *Comput. Vis. Image Und.*, vol. 77, no. 2, pp. 192–210, Feb. 2000.

[13] T. Rohlfing, R. Brandt, C. R. Maurer Jr., and R. Menzel, "Bee brains, B-splines and computational democracy: Generating an average shape atlas," in *Proc. IEEE Workshop Math. Methods Biomed. Image Anal.*, L. Staib, Ed., Kauai, HI, 2001, pp. 187–194.

[14] S. K. Warfield, F. Jolesz, and R. Kikinis, "A high performance approach to the registration of medical imaging data," *Parallel Comput.*, vol. 24, no. 9–10, pp. 1345–1368, 1998.

[15] G. E. Christensen, M. I. Miller, M. W. Vannier, and U. Grenander, "Individualizing neuroanatomical atlases using a massively parallel computer," *IEEE Comput.*, vol. 29, no. 1, pp. 32–38, Jan. 1996.

[16] S. K. Warfield, A. Nabavi, T. Butz, K. Tuncali, S. G. Silverman, P. M. Black, F. A. Jolesz, and R. Kikinis, "Intraoperative segmentation and nonrigid registration for image guided therapy," in *Proc. 3rd Int. Conf. Medical Image Comput. Comput.-Assisted Intervention*, vol. 1935, Lecture Notes in Computer Science, S. L. Delp, A. M. DiGoia, and B. Jaramaz, Eds. Berlin, Germany, 2000, pp. 176–185.

[17] G. E. Christensen, S. Joshi, and M. Miller, "Volumetric transformation of brain anatomy," *IEEE Trans. Med. Imag.*, vol. 16, pp. 864–877, Dec. 1997.

[18] C. Davatzikos, "Spatial transformation and registration of brain images using elastically deformable models," *Comput. Vis. Image Und.*, vol. 66, no. 2, pp. 207–222, May 1997.

[19] J.-P. Thirion, "Image matching as a diffusion process: An analogy with Maxwell's demons," *Med. Image Anal.*, vol. 2, no. 3, pp. 243–260, 1998.

[20] D. Rueckert, "Nonrigid registration: Concepts, algorithms, and applications," in *Medical Image Registration*, J. V. Hajnal, D. L. G. Hill, and D. J. Hawkes, Eds. Boca Raton, FL: CRC, 2001, pp. 281–301.

[21] A. W. Toga, Ed., *Brain Warping*. San Diego, CA: Academic, 1998.

[22] H. Lester and S. R. Arridge, "A survey of nonlinear medical image registration," *Pattern Recognit.*, vol. 32, no. 1, pp. 129–149, Jan. 1999.

[23] A. Geist, A. Beguelin, J. Dongarra, J. Weicheng, R. Manchek, and V. Sunderam, *PVM: Parallel Virtual Machine*. Cambridge, MA: MIT Press, 1994.

[24] B. Lewis and D. J. Berg, *Threads Primer: A Guide to Multithreaded Programming*. Upper Saddle River, NJ: Prentice-Hall, 1996.

[25] C. Studholme, D. L. G. Hill, and D. J. Hawkes, "An overlap invariant entropy measure of 3D medical image alignment," *Pattern Recognit.*, vol. 33, no. 1, pp. 71–86, 1999.

[26] F. Maes, A. Collignon, D. Vandermeulen, G. Marchal, and P. Suetens, "Multimodality image registration by maximization of mutual information," *IEEE Trans. Med. Imag.*, vol. 16, pp. 187–198, Apr. 1997.

[27] C. Studholme, D. L. G. Hill, and D. J. Hawkes, "Automated three-dimensional registration of magnetic resonance and positron emission tomography brain images by multiresolution optimization of voxel similarity measures," *Med. Phys.*, vol. 24, no. 1, pp. 25–35, Jan. 1997.

[28] T. W. Sederberg and S. R. Parry, "Free-form deformation and solid geometric models," *Comput. Graph.*, vol. 20, no. 4, pp. 151–160, 1986.
[29] S. Lee, G. Wolberg, and S. Y. Shin, "Scattered data interpolation with multilevel B-splines," *IEEE Trans. Visualization Comput. Graph.*, vol. 3, no. 3, pp. 228–244, July 1997.
[30] J. A. Nelder and R. Mead, "A simplex method for function minimization," *Comput. J.*, vol. 7, pp. 308–313, 1965.
[31] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge, UK: Cambridge Univ. Press, 1988.
[32] G. M. Amdahl, "Validity of the single-processor approach to achieving large scale computing capabilities," in *Proc. AFIPS Conf.*, vol. 30, Reston, VA, 1967, pp. 483–485.
[33] P. M. Black, T. Moriarty, E. Alexander III, P. Stieg, E. J. Woodward, P. L. Gleason, C. H. Martin, R. Kikinis, R. B. Schwartz, and F. A. Jolesz, "Development and implementation of intraoperative magnetic resonance imaging and its neurosurgical applications," *Neurosurgery*, vol. 41, pp. 831–845, 1997.
[34] V. M. Tronnier, C. R. Wirtz, M. Knauth, G. Lenz, O. Pastyr, M. M. Bonsanto, F. K. Albert, R. Kuth, A. Staubert, W. Schlegel, K. Sartor, and S. Kunze, "Intraoperative diagnostic and interventional magnetic resonance imaging in neurosurgery," *Neurosurgery*, vol. 40, pp. 891–902, 1997.

Torsten Rohlfing received the M.Sc. degree in computer science from the University of Karlsruhe, Karlsruhe, Germany, in 1997 and the Ph.D. degree in engineering/computer science from the Technical University of Berlin, Berlin, Germany, in 2000.

He is currently a Postdoctoral Research Fellow with the Image Guidance Laboratories, Department of Neurosurgery, Stanford University, Stanford, CA. His research interests include nonrigid image registration methods and their clinical applications, high-performance computing, small animal imaging, and registration of three-dimensional tomographic images to two-dimensional X-ray projection images for intraoperative guidance.

Calvin R. Maurer, Jr. (S'96–M'96) received the B.S.E. degree in chemical engineering from Princeton University, Princeton, NJ, and the M.S. and Ph.D. degrees in biomedical engineering from Vanderbilt University, Nashville, TN.

He was a Postdoctoral Fellow in the Computational Imaging Science Group, Guy's Hospital, King's College, London, U.K., and an Assistant Professor of Neurosurgery, Biomedical Engineering, and Radiation Oncology at the University of Rochester, Rochester, NY. He is currently a Consulting Assistant Professor and Co-Director of the Image Guidance Laboratories, Stanford University, Stanford, CA. He has authored approximately 100 publications, including 30 peer-reviewed journal papers. His current research interests include image registration, data fusion, visualization, tissue deformation, and shape representation, with applications to image-guided therapy, augmented reality in surgery, 3-D ultrasound, and interventional MR imaging.