A Data Distributed Parallel Algorithm for Nonrigid Image Registration [⋆]

Fumihiko Ino [a,∗], Kanrou Ooyama [b], Kenichi Hagihara [a]

[a] *Graduate School of Information Science and Technology, Osaka University, 1-3 Machikaneyama, Toyonaka, Osaka 560-8531, Japan*

[b] *Matsushita Electric Industrial Co., Ltd. 1006 Kadoma, Kadoma, Osaka 571-8501, Japan*

Abstract

Image registration is a technique for defining a geometric relationship between each point in images. This paper presents a data distributed parallel algorithm that is capable of aligning large-scale three-dimensional (3-D) images of deformable objects. The novelty of our algorithm is to overcome the limitations on the memory space as well as the execution time. In order to enable this, our algorithm incorporates data distribution, data-parallel processing, and load balancing techniques into Schnabel's registration algorithm that realizes robust and efficient alignment based on information theory and adaptive mesh refinement. We also present some experimental results obtained on a 128-CPU cluster of PCs interconnected by Myrinet and Fast Ethernet switches. The results show that our algorithm requires less amount of memory resources, so that aligns datasets up to $1024 \times 1024 \times 590$ voxel images with reducing the execution time from hours to minutes, a clinically compatible time.

Key words: Nonrigid image registration; Adaptive mesh refinement; Free-form deformation; Data distribution; Load balancing

1 Introduction

Image registration [1,2] is the process of matching images so that defines a geometric relationship between each point in the images. This technique is increasing its

[⋆] This work was partly supported by JSPS Grant-in-Aid for Scientific Research on Priority Areas (16016254 and 16035209) by the Ministry of Education, Culture, Sports, Science and Technology of Japan.

[∗] Corresponding author.
Email address: ino@ist.osaka-u.ac.jp (Fumihiko Ino).

role in medical diagnosis with the rapid advances in radiologic imaging such as X-ray computed tomography (CT) and magnetic resonance (MR) scans. For instance, it assists surgeons by relating preoperative images to intraoperative images during image-guided surgery [3–5], creates novel multimodality images by combining information from different modality images [6–8], and helps medical doctors in detecting cancers by monitoring changes in size, shape, or image intensity over time intervals [9, 10].

Registration algorithms can be classified into two groups whether they intend to align the rigid body or the nonrigid body. Rigid registration relates images by a rigid transformation with 6 degrees of freedom (DOF), representing rotations and translations. In order to correct the influences of image distortions such as scaling and skewing errors, some algorithms are further parameterized by an affine transformation, which has additional 6 DOF to enable scaling and shearing of images. Although 12 DOF are sufficient for rigid registration, more DOF are required for nonrigid registration, which addresses deformable objects. For instance, a B-spline free-form deformation (FFD) [9] defined by a $10 \times 10 \times 10$ mesh of control points yields a transformation with 3000 DOF [9], so that successfully aligns images of deformable objects such as brains [4,5,11,12], breasts [5,9,12], and livers [10,11]. However, these many DOF increase the amount of computation required for nonrigid registration tasks. For example, our preliminary experiments show that aligning $512 \times 512 \times 295$ voxel images of the liver takes approximately 12 hours on a single Pentium III 1-GHz system. This long execution time is unacceptable to image-guided surgery, where the execution time must be less than 10 minutes to reduce patients physical load such as blood transfusions. Therefore, in order to achieve intraoperative nonrigid registration, one major challenging problem is to reduce its execution time without degrading the accuracy of alignments and the resolution of images.

Earlier projects tackle this problem by employing high performance computing (HPC) approach with shared memory architectures [4, 5, 13, 14] and distributed memory architectures [4, 13–17]. These projects successfully demonstrate the time benefits of HPC. However, except for Hastings's pipeline algorithm [17], all of their algorithms are lacking the capability of data distribution, so that every processor holds the entire 3-D images during registration process. Therefore, the image size available on a HPC system is strictly bounded by the memory space of a node in the system. This strict restriction is undesirable for registration of high-resolution images because on-memory computation is essential to enable fast registration. For example, aligning two $1024 \times 1024 \times 1024$ voxel datasets, each represented in 16-bit color depth, requires at least 4 GB of physical memory to carry out on-memory computation.

The purpose of this paper is to demonstrate the space benefits of HPC as well as the time benefits. To do this, we present a data distributed parallel algorithm that is capable of large-scale image registration with scalable performance on distributed memory multiprocessor systems. Our algorithm parallelizes Schnabel's registration algorithm [11] that enables robust multimodality registration with no user interaction and preprocessing. Furthermore, our parallel algorithm uses Rohlfing's numerical optimization [5] with the following three techniques.

Data distribution: Data distribution enables us to increase the data size available on a HPC system by assigning a portion of 3-D images to processors.

Data-parallel processing: Data-parallel processing realizes fast registration by allowing processors to independently process the assigned portion in parallel.

Load balancing: Load balancing improves parallel efficiency by balancing processor workloads imbalanced due to the difference of computational costs associated with each portion.

The remainder of the paper is organized as follows. Section 2 introduces some related work on parallelization of image registration. Section 3 presents an overview of the sequential algorithm proposed by Schnabel et al. and Section 4 describes the design and implementation of our algorithm that parallelizes the sequential algorithm. Section 5 shows experimental results on a 128-CPU cluster of PCs. Finally, Section 6 concludes the paper.

2 Related Work

Several recent works employ HPC approach for image registration. Warfield et al. [14] present a parallel nonrigid algorithm based on the workpile paradigm [18], in which one manager process manages a pool of tasks while the remaining worker processes request a task and independently perform the task assigned from the manager. On a cluster of two Sun Enterprise Server 5000s each with eight 167 MHz CPUs, their algorithm obtain intrapatient and interpatient registrations of $256 \times 256 \times 52$ voxel images in less than 10 minutes, which is a clinically compatible range. They also present another algorithm [4] that takes account of a biomechanical model of the brain. This algorithm realizes intraoperative registration of the brain within 10 seconds on a Sun Ultra HPC 6000 with twenty 250 MHz CPUs. Although it employs a static load balancing strategy with equally sized domains being assigned to each CPU, the speedup for 20 CPUs is approximately a factor of 8, so that the load imbalance issue remains unaddressed. Nevertheless, their HPC approach has successfully assisted surgeons during surgery.

Christensen [13] compares registration algorithms implemented on multiple instruction multiple data (MIMD) and single instruction multiple data (SIMD) computers. In this comparison, a 16-CPU SGI Challenge, a SIMD computer, show a nearly linear speedup for $128 \times 128 \times 100$ voxel images. In contrast, a 128×128 mesh connected MasPar MP-2, a MIMD computer, suffers a loss in efficiency because the nonlinear nature of the deformations requires random access to the whole memory, reducing data transfer parallelism on the mesh-connected network.

Rohlfing et al. [5] employ a numerical optimization technique as well as HPC approach. Their method accelerate nonrigid registration for brain MR images of 256 $\times$ 256 $\times$ 100 voxels from an hour on a single CPU system to approximately a 100 seconds on a 64-CPU SGI Origin 3800 running at 400 MHz. Because this multiprocessor system offers a shared address space, it allows existing sequential algorithms to be converted easily for parallel execution. However, shared memory systems are much expensive compared to distributed memory systems such as clusters of PCs, which mainly consist of off-the-shelf components [19, 20].

Ourselin et al. [15] demonstrate affine registration on a 20-CPU cluster of off-the-shelf symmetric multiprocessor (SMP) PCs, yielding a speedup of a factor of 8. Their implementation is based on a combination of the message passing paradigm [21] and the shared memory paradigm [22] for inter-node and intra-node communication, respectively. Butz et al. [16] also present parallel affine registration based on genetic optimization. Their master/slave implementation on a 20-CPU cluster of Pentium III 550-MHz PCs takes approximately 30 minutes for 256 $\times$ 256 $\times$ 123 voxel images.

Another emerging HPC infrastructure is the Grid [23, 24], where HPC resources are dynamically shared among virtual organizations over the Internet. Hastings et al. show a toolkit [17] that allows rapid and efficient development of image segmentation and registration applications in a distributed environment. To deal with a time series of 3-D images, their toolkit exploits task and data parallelism in a chain of processing operations that begins with data acquisition and ends with data visualization. They employ a data-stream programming model, in which the data flows through pipeline stages, each accelerated by data-parallel processing. Because every stage processes a portion of the data, this model relaxes the requirement of the memory space at each stage. It achieves full acceleration if every pipeline stage takes the same time and data transfer between the stages takes much shorter time than data-parallel processing at each stage. They are currently developing interfaces to the Globus Toolkit [25], which provides standard Grid mechanisms such as resource, security, and file management.

In contrast to the above earlier works that mainly demonstrate the time benefits of HPC, the novelty of the paper is to reduce the memory usage per node in a data-parallel programming model, aiming to demonstrate the space benefits of HPC. Note here that the data-stream programming model also reduces the memory usage per node. However, to take this advantage, the entire data must be decomposed into portions such that each portion can be locally processed at every pipeline stage. Moreover, we also have to form the registration algorithm in a pipeline such that every pipeline stage takes the same processing time to obtain full acceleration. For time series images, this could easily be realized by streaming each time step image through the stages, because in many cases, different images produced at different time could independently be processed at the same time. However, it is not easy for 3-D images, because the deformations in the 3-D space prevent us from decomposing them into small portions, for example, 2-D slices or 3-D blocks. Such decompositions are available only if the deformations of each portion do not influence those of others. Therefore, we think that the data-parallel programming model is more flexible with 3-D image registration, as compared to the data-stream programming model.

3 Nonrigid Image Registration

For better understanding of our parallel algorithm, this section briefly summarizes overviews of Schnabel's algorithm and Rohlfing's optimization presented fully in [11] and in [5], respectively.

Let F and R be the floating and reference images, respectively. In order to register images F and R, Schnabel's algorithm computes a nonrigid transformation that optimizes function $\mathcal{C}$, a cost function associated with a voxel-based similarity measure between F and R. This algorithm has three advantages as follows:

- Hierarchical, locally controlled FFDs [9] by B-spline functions [26];
- A robust similarity measure [27] by normalized mutual information (NMI);
- An efficient registration by adaptive mesh refinement [11].

3.1 B-spline Free-Form Deformations (FFDs)

One familiar technique to represent a nonrigid transformation is to employ spline functions such as B-splines [26], thin-plate splines [28], and elastic-body splines [29]. Because B-splines are locally controlled, they are computationally efficient compared to the other globally controlled splines.

As illustrated in Fig. 1(a), a B-spline FFD [9] represents a nonrigid transformation by manipulating a mesh of control points arranged in the image domain, $\Omega = \{(x, y, z) \mid 0 \leq x < X, 0 \leq y < Y, 0 \leq z < Z\}$. Let Φ be a 3-D mesh of control points and $\mathbf{T} : (x, y, z) \mapsto (x', y', z')$ be a transformation of any point (x, y, z) in image F to its corresponding point (x', y', z') in image R. Given a mesh of control points $\phi_{i,j,k}$ with uniform spacing δ mm, nonrigid transformation $\mathbf{T}$ by B-spline functions is defined by

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

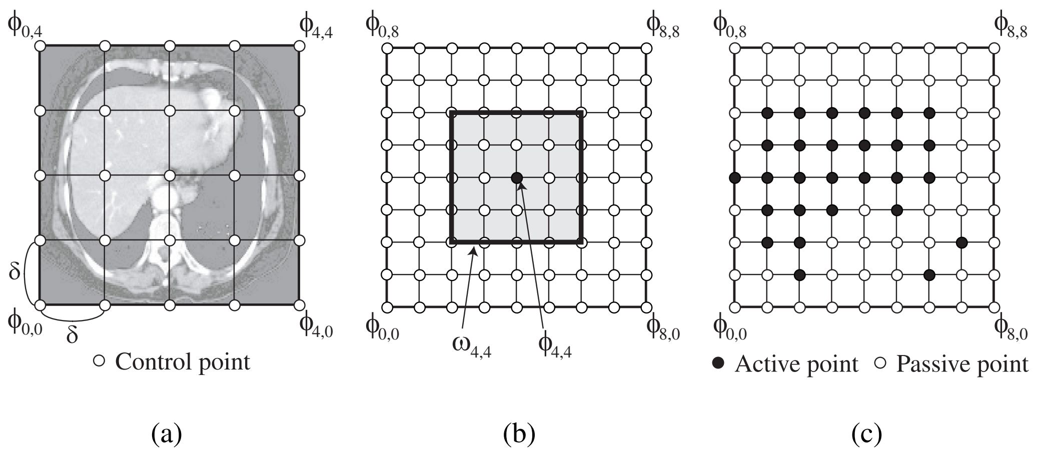


Fig. 1. Hierarchical B-spline free-form deformations (FFDs) based on adaptive mesh refinement. (a) Deformations of a floating image are performed by manipulating an overlaying mesh of control points in a coarse to fine fashion. δ represents the spatial resolution of control points and $\phi_{i,j}$ represents the control point located on the i-th column of the j-th row. (b) Control point $\phi_{i,j}$ affects points only inside its $4\delta \times 4\delta$ neighborhood domain $\omega_{i,j}$. (c) During these deformations, only active control points are allowed to move while passive control points stay fixed.

where $i = \lfloor x/\delta \rfloor - 1$, $j = \lfloor y/\delta \rfloor - 1$, $k = \lfloor z/\delta \rfloor - 1$, $u = x/\delta - \lfloor x/\delta \rfloor$, $v = y/\delta - \lfloor y/\delta \rfloor$, $w = z/\delta - \lfloor z/\delta \rfloor$, and B_l represents the l-th basis function of cubic B-splines. Eq. (1) indicates that B-spline FFDs are locally controlled because the deformation of any point (x, y, z) is determined by its surrounding $4 \times 4 \times 4$ neighborhood of control points. In other words, as shown in Fig. 1(b), each control point $\phi_{i,j,k}$ affects only its $4\delta \times 4\delta \times 4\delta$ neighborhood domain $\omega_{i,j,k}$, a subdomain of Ω. Furthermore, by organizing mesh Φ and images F and R in a hierarchy, the B-spline FFD represents a wide range of deformations with a lower computational complexity. Thus, Schnabel's algorithm aligns images in a coarse to fine manner, where the spatial resolutions of images and control points, γ and δ, respectively, are decreased at each level of hierarchy, namely deformation level.

Note here that Eq. (1) is computed for all points (x, y, z) in Ω by a triple nested zyx-loop, and moreover, it can be rewritten as

$$\mathbf{T}(x, y, z) = \sum_{l=0}^{3} B_l(u)\hat{\phi}_{i+l}, \tag{2}$$

where $\hat{\phi}_{i+l} = \sum_{m=0}^{3}\sum_{n=0}^{3} B_m(v)B_n(w)\phi_{i+l,j+m,k+n}$, as Rohlfing et al. [5] pointed out. Eq. (2) indicates that the computed result of $\hat{\phi}_{i+l}$ is the same for all points (x, y, z) in one row located within the same cell of the mesh. Therefore, moving this computation outside the most inner x-loop reduces the total computational cost for the triple nested loop.

The second advantage of Schnabel's algorithm is a similarity measure based on information theory, which realizes robust registration of multimodality images and contrast-enhanced images [5,7,9–12,30]. This similarity measure is based on NMI proposed by Studholme et al. [27] who have been experimentally confirmed that NMI is more robust than mutual information proposed by Maes et al. [7].

NMI represents the amount of information that one image, A, contains about a second image, B. The similarity measure between images A and B is given by

$$\mathcal{C}_{\text{similarity}}(A, B) = \frac{H(A) + H(B)}{H(A, B)}, \tag{3}$$

where $H(A)$ and $H(B)$ are the marginal entropies of A and B, and $H(A, B)$ is their joint entropy, each given by

$$H(A) = -\sum_{a} p_A(a) \log p_A(a), \tag{4}$$

$$H(A, B) = -\sum_{a,b} p_{AB}(a, b) \log p_{AB}(a, b), \tag{5}$$

where $p_A(a)$ and $p_{AB}(a, b)$ are the marginal and joint probability distributions of the intensity values, respectively. Both distributions are obtained by normalizing the two-dimensional (2-D) joint histogram $h(A, B)$ of A and B. When A and B are matched, NMI is maximized because A explains B most effectively.

By using nonrigid transformation $\mathbf{T}$ and similarity measure $\mathcal{C}_{\text{similarity}}$, the cost function for optimization is defined as follows:

$$\mathcal{C}(\Phi) = -\mathcal{C}_{\text{similarity}}(R, \mathbf{T}(F)). \tag{6}$$

In order to find the optimal transformation parameter Φ that minimizes Eq. (6), the algorithm employs the steepest descent optimization [31] during registration process. The algorithm stops this optimization if a local optimum has been found. Here, it assumes a local optimum if $||\nabla\mathcal{C}|| \leq \epsilon$, where $||\nabla\mathcal{C}||$ and ϵ represent the gradient norm of cost function $\mathcal{C}$ and a threshold for minimization, respectively. The gradient $\nabla\mathcal{C}$ of the cost function is estimated by using the finite-difference approximation [31]. As we mentioned in Section 3.1, this optimization is performed from coarse to fine resolution by traveling deformation levels.

3.3 Adaptive Mesh Refinement

Adaptive mesh refinement [11,30,32] is a technique to further reduce the computational complexity given by hierarchically organized B-spline FFDs. This technique allows control points to move only where deformations need to be modeled.

To enable this, Schnabel's algorithm associates each control point $\phi_{i,j,k}$ with a status, $\mathcal{S}(\phi_{i,j,k}) \in \{\text{active}, \text{passive}\}$, as shown in Fig. 1(c). During registration process, active control points are allowed to move while passive control points stay fixed. Thus, only active control points are parameters of cost function $\mathcal{C}$, so that the algorithm minimizes $\mathcal{C}(\Phi^+)$, where Φ^+ represents a mesh of active control points.

Status $\mathcal{S}(\phi_{i,j,k})$ is given by

$$\mathcal{S}(\phi_{i,j,k}) = \begin{cases} \text{active}, & \text{if } \mathcal{M}(\omega_{i,j,k}) > \alpha, \\ \text{passive}, & \text{otherwise}, \end{cases} \tag{7}$$

where $\mathcal{M}$ represents a statistical measure and α represents a threshold for selection. The statistical measures employed in Schnabel's algorithm are as follows.

- *Reference image measures* for excluding image background regions at the beginning of deformation levels. They propose two reference image measures: the Shannon-Wiener entropy H and intensity variance σ.
- *Joint image pair measures* for describing the degree of image alignment after each deformation. They propose a consistent, generalized measure based on the local gradient $\partial\mathcal{C}/\partial\phi_{i,j,k}$ of cost function $\mathcal{C}$.

We experimentally determined the following two measures for our parallel algorithm. For reference image measures, we use $H(\omega_{i,j,k})/H(R)$, where $H(\omega_{i,j,k})$ is the local image entropy computed over subdomain $\omega_{i,j,k}$. For joint image pair measures, we use $||\partial\mathcal{C}/\partial\phi_{i,j,k}||/||\partial\mathcal{C}/\partial\Phi^+||$, where $||\partial\mathcal{C}/\partial\phi_{i,j,k}||$ represents the norm of local gradient $\partial\mathcal{C}/\partial\phi_{i,j,k}$.

4 Parallelizing Nonrigid Image Registration

This section describes why we use the three techniques mentioned in Section 1 and how we incorporate them into our algorithm. Fig. 2 summarizes our parallel algorithm.

```
Parallel nonrigid registration algorithm

initialize $\gamma$ and $\delta$, the spatial resolution of images and that of control points, respectively.
repeat
    load a responsible portion of images $F$ and $R$ with $\gamma$.        // Data distribution
    initialize control points $\Phi^+$ and each status $\mathcal{S}$ by a reference image measure.
    repeat
        compute in parallel the gradient vector of the cost function in Eq. (6) with
        respect to nonrigid transformation parameters $\Phi^+$:
            $\nabla\mathcal{C} = \partial\mathcal{C}/\partial\Phi^+$.               // Gradient computation
        update $\Phi^+$ and $\mathcal{S}$ by a joint image pair measure.
        compute in parallel the cost function $\mathcal{C}(\Phi^+)$.      // Similarity computation
        while $\mathcal{C}(\Phi^+)$ approaches to the optimum do
            compute control points $\Phi^+ = \Phi^+ + \nabla\mathcal{C}/||\nabla\mathcal{C}||$.
            compute in parallel $\mathcal{C}(\Phi^+)$.                    // Similarity computation
    until $||\nabla\mathcal{C}|| \leq \epsilon$.
    increase deformation level $L$ by decreasing spatial resolutions $\gamma$ and $\delta$.
until $\gamma$ and $\delta$ reach the finest resolution.
```

Fig. 2. Parallel nonrigid registration algorithm based on Schnabel's algorithm [11]. Our parallel algorithm mainly consists of data distribution, gradient computation, and similarity computation. This pseudo code uses Rueckert's representation [9].

4.1 Performance Analysis of Sequential Algorithm

Before describing the details of our parallel algorithm, we first present the preliminary performance analysis of the sequential algorithm. The algorithm has two performance bottlenecks: *the gradient computation* required for the steepest descent optimization and *the similarity computation* required for the cost function evaluation, accounting for 92% and 7% of total execution time, respectively.

In the gradient computation, the algorithm computes local gradient $\partial\mathcal{C}/\partial\phi_{i,j,k}$ for each active control point $\phi_{i,j,k}$. This gradient can be computed from small $4\delta \times 4\delta \times 4\delta$ neighborhood subdomain $\omega_{i,j,k}$, because B-splines are locally controlled. However, the gradient computation becomes a performance bottleneck at finer deformation levels, because the total number of control points significantly increases after mesh refinement.

On the other hand, the similarity computation mainly consists of the construction of 2-D joint histogram $h(R, \mathbf{T}(F))$. Although this construction seems to be a small bottleneck, it also has to be parallelized in order to obtain scalable performance on our cluster with more than 100 processors. Otherwise, as Amdahl's law [33] says, the speedup for N processors is bounded by a factor of $\lim_{N\to\infty}(100/(7 + 92/N)) < 15$. That is, although many processors are available, the speedup never reaches a factor of 15 if the similarity computation is processed in sequential.

According to the above analysis, our algorithm parallelizes both the gradient computation and the similarity computation in order to enable large-scale registration with scalable performance.

4.2 Data Distribution

In the sequential algorithm, 3-D images F and R occupy the most of memory space compared to the remaining data structures such as joint histogram $h(R, \mathbf{T}(F))$ and transformation parameters Φ. Therefore, we focus on how to distribute F and R with less performance loss. Note here that data distribution is necessary only at fine resolution levels where on-memory computation is unavailable due to the lack of physical memory. At coarse resolution levels where the image size is small enough due to large sampling rate γ, our algorithm uses no data distribution scheme to obtain higher performance with less communication.

Our algorithm uses a block distribution in order to achieve higher speedup for the hotspot of the sequential algorithm. As we mentioned in Section 4.1, the hotspot is the gradient computation, where each active control point $\phi_{i,j,k}$ requires neighborhood block domain $\omega_{i,j,k}$ to compute local gradient $\partial\mathcal{C}/\partial\phi_{i,j,k}$. Therefore, although cyclic distributions enable statically load balanced computation, we avoid using them because they significantly decrease the speedup due to frequent communication caused by this data access pattern.

We now describe how the algorithm determines the block size for N processors. An appropriate block size depends on the tradeoff between the memory usage and the execution time. Maximizing the block size corresponds to no data distribution, which achieves the best performance with less communication but needs many memory resources at every processor. In contrast, minimizing the block size corresponds to disjoint block distribution (Fig. 3(a)), which realizes the minimum usage of memory resources but requires many communication due to the gradient computation computed from neighborhood domain $\omega_{i,j,k}$.

Our approach is to select a balancing point of this tradeoff. As illustrated in Fig. 3(b), the algorithm divides the images into partially overlapped blocks with marginal length l in order to prevent communication during data-parallel processing of performance bottlenecks: the gradient computation and the similarity computation. In the following discussion, let Ω_s be a disjoint block of domain Ω in which processor P_s takes responsibility for computation, where $1 \leq s \leq N$ and s represents a processor id. Let F_s and R_s also denote subimages of F and R that processor P_s loads, respectively. For all $1 \leq s \leq N$, the algorithm satisfies the following two conditions to select the balancing point.

- C1: Condition for the gradient computation.
  For all active control points $\phi_{i,j,k}$ in subdomain Ω_s, subimage F_s includes $4\delta \times 4\delta \times 4\delta$ neighborhood domain $\omega_{i,j,k}$ and subimage R_s includes transformed

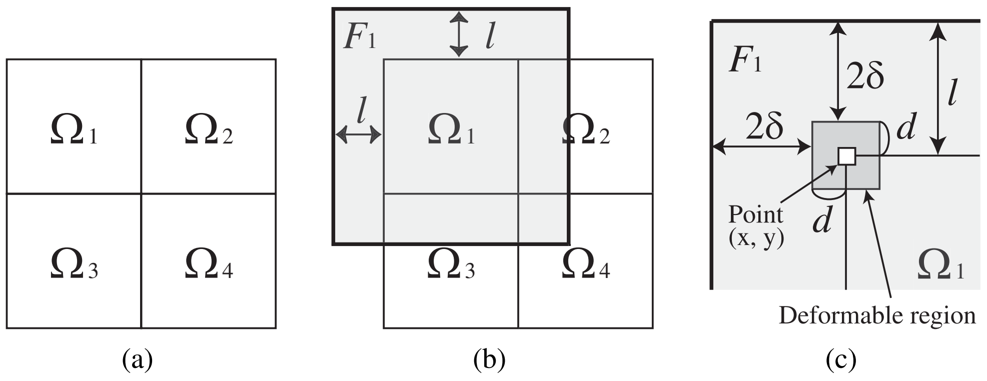


Fig. 3. Image distribution for 4 processors. (a) Disjoint block distribution, (b) partially overlapped block distribution with marginal length l, and (c) its zooming view. For all $1 \le s \le N$, processor P_s holds subimage F_s and takes responsibility for computation in subdomain Ω_s. Each point is allowed to move within $2d \times 2d$ deformable region.

neighborhood domain $\mathbf{T}(\omega_{i,j,k})$.

- C2: Condition for the similarity computation.
  For all points (x, y, z) in subdomain Ω_s, subimage F_s includes (x, y, z) and subimage R_s includes transformed point $\mathbf{T}(x, y, z)$.

The above conditions C1 and C2 allow every processor P_s to locally compute every gradient $\partial\mathcal{C}/\partial\phi_{i,j,k}$ in each responsible block Ω_s and to locally create a joint histogram for Ω_s, respectively.

An appropriate marginal length for satisfying both conditions C1 and C2 is determined as follows. Given the maximum length of correct deformations in v mm, marginal length l is given by

$$l = d + 2\delta \quad \text{such that} \quad d \ge v, \tag{8}$$

where d is the length for deformable region in mm, as shown in Fig. 3(b). Eq. (8) satisfies condition C1 because $l \ge v + 2\delta$. It also satisfies condition C2 because $l > v$.

Eq. (8) assumes that the correct length v is known before registration. Although this precise length is usually unknown, we think that the algorithm is acceptable to clinical use, because a typical value estimated roughly from users' experiences can be substituted for v.

4.3 Parallel Gradient Computation

Because adaptive mesh refinement causes active control points in a non-uniform distribution, the processor workloads associated with each subdomain Ω_s become imbalanced in the compute-intensive gradient computation. To address this load imbalance issue, we first analyzed the distribution characteristics of active control points by using liver images (Fig. 1(a)). We then found that the workloads become more imbalanced as N increases. For example, only a quarter of 128 processors are responsible for more than 80% of active control points. The remaining processors have few active and many passive control points, because their responsible subdomain corresponds mainly to rigid objects, such as the bone and image background, which rapidly complete the alignment compared to nonrigid objects.

To give an answer to this issue, our algorithm employs two load balancing strategies. One is for coarse resolution levels where data distribution is unnecessary and the other is for fine resolution levels where data distribution is necessary to carry out on-memory computation.

For coarse resolution levels, since every processor holds the entire images and knows which control points are active, we simply assign active control points to processors in a round-robin manner.

For fine resolution levels, our load balancing strategy uses a greedy algorithm to place processors into groups such that a group consists of at least one high-loaded processor and some low-loaded processors. This strategy aims to balance the workloads within the same group, enabling less communication by preventing processors from communicating between different groups. Note here that our algorithm is similar to Graham's list scheduling algorithm [34], a greedy algorithm to schedule tasks to processors. That is, tasks and processors in Graham's algorithm are regarded as processors and groups of processors in our algorithm, respectively.

Let Φ_s^+ be a set of active control points in subdomain Ω_s, where $1 \le s \le N$. Let M denote the number of high-loaded processors defined as processors with more than W active control points, where $W = \sum_{s=1}^{N} |\Phi_s^+|/N$, representing the number of active control points per processor. Then, our algorithm classifies processors into groups in the following two phases.

(1) Group initialization: The algorithm creates M groups of processors, G_1, G_2, ..., G_M, each initialized with a different one of the M high-loaded processor.
(2) Group construction: The algorithm create a list, L, in which the remaining $N - M$ low-loaded processors are sorted by $|\Phi_s^+|$ in an ascending order. Then, it repeats the following operations until L becomes empty: Select group G_t such that $W(G_t)$ is the maximum, where $1 \le t \le M$ and $W(G_t) = \sum_{P_s \in G_t} |\Phi_s^+|/|G_t|$, representing the number of active control points per processor in G_t; Delete a processor from the head of L and add it to G_t.

After the above phases, the algorithm obtains M groups, each represented as $G_t = \{P_{t,1}, P_{t,2}, \ldots, P_{t,|G_t|}\}$, containing one high-loaded processor $P_{t,1}$ and some low-loaded processors $P_{t,2}, \ldots, P_{t,|G_t|}$ (if any), where $1 \le t \le M$.

We now describe how the algorithm performs load balancing for given group G_t, where $1 \leq t \leq M$. To enable this, low-loaded processors $P_{t,2}, \ldots, P_{t,|G_t|}$ have to share the data assigned to high-loaded processor $P_{t,1}$. Such data contains subimages $F_{t,1}$ and $R_{t,1}$, and information on active control points $\Phi^+_{t,1}$. In addition to phases (1) and (2), the following phases perform the parallel gradient computation with load balancing capability.

(3) Data redistribution: For all $1 \leq t \leq M$, high-loaded processor $P_{t,1}$ multicasts subimages $F_{t,1}$ and $R_{t,1}$, and active control points $\Phi^+_{t,1}$ to the remaining low-loaded processors $P_{t,2}, \ldots, P_{t,|G_t|}$.
(4) Workload distribution: For all $1 \leq t \leq M$, active control points $\Phi^+_{t,1}$ are disjointly decomposed into $\lambda_{t,1}, \lambda_{t,2}, \ldots, \lambda_{t,|G_t|}$ such that $|\lambda_{t,1}| = W(G_t)$ and $|\lambda_{t,u} \cup \Phi^+_{t,u}| = W(G_t)$, for all $2 \leq u \leq |G_t|$.
(5) Gradient computation: For all $1 \leq t \leq M$, processors $P_{t,1}$ and $P_{t,u}$, where $2 \leq u \leq |G_t|$, independently compute gradients for $\lambda_{t,1}$ and $\lambda_{t,u} \cup \Phi^+_{t,u}$, respectively.
(6) Result distribution: $P_{t,1}$ gathers the computation results from $P_{t,2}$, ..., $P_{t,|G_t|}$, and then broadcasts them to all N processors.

4.4 Parallel Similarity Computation

In order to accelerate the similarity computation, our algorithm parallelizes the construction of the 2-D joint histogram by means of the binary-swap (BS) method [35]. The BS method is originally proposed to generate a single image by compositing N locally rendered images. This computation is similar to the similarity computation, where processors locally construct joint histograms $h_1, h_2, \ldots, h_N$ for their responsible subdomain $\Omega_1, \Omega_2, \ldots, \Omega_N$, respectively, and then merge them into a global joint histogram $h(R, \mathbf{T}(F))$ for the entire domain Ω. One advantage of BS is highly scalable performance produced by data-parallel processing and tree structured merging. As illustrated in Fig. 4, it finishes merging the local images in $\log N$ stages, in which all N processors participate in performing tasks.

Our algorithm produces joint histogram $h(R, \mathbf{T}(F))$ in the following two phases.

(a) Local histogram construction: For all $1 \leq s \leq N$, processor P_s independently creates local joint histogram h_s for its responsible subdomain Ω_s.
(b) Binary-swap merging: All created local histograms $h_1, h_2, \ldots, h_N$ are merged into the global histogram $h(R, \mathbf{T}(F))$ by the BS method. As illustrated in Fig. 4, every processor is paired up. Every pair of processors splits its local histogram into two pieces, and each processor takes responsibility for one of the two pieces. Repeating this splitting and exchanging with different pairs for $\log N$ stages generates the global histogram $h(R, \mathbf{T}(F))$ in a distributed manner.

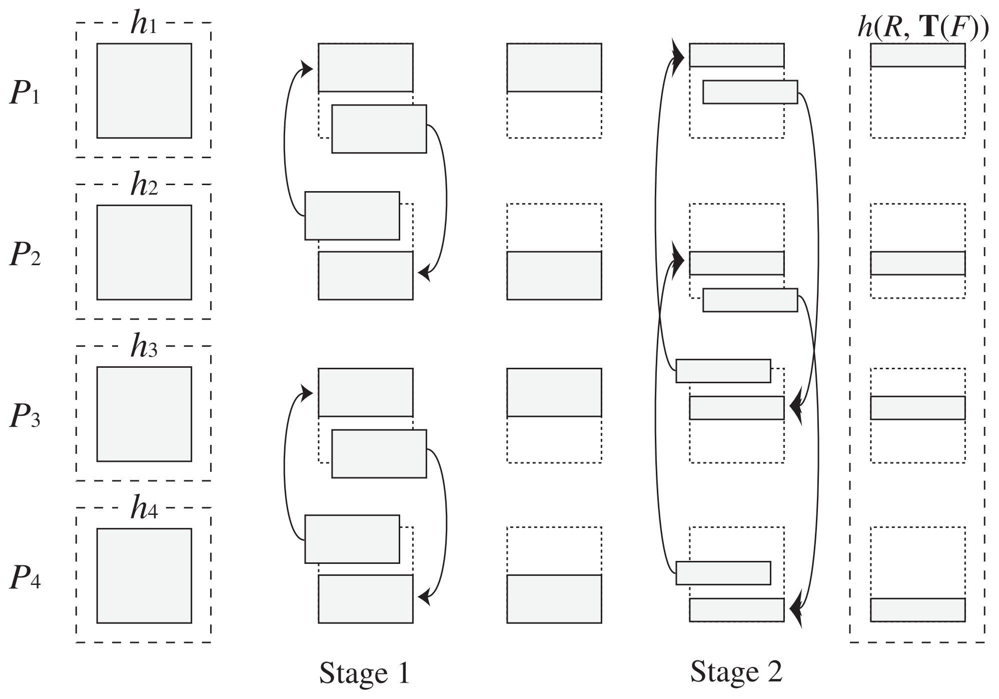


Fig. 4. Binary-swap (BS) method applied to joint histogram construction on 4 processors. $h_1, \ldots, h_4$ represent local joint histograms computed from each responsible subdomain. On N processors, each processor has $1/N$ portion of the joint histogram $h(R, \mathbf{T}(F))$ after $\log N$ stages.

We now describe how our algorithm computes the similarity measure $\mathcal{C}_{\text{similarity}}(A, B)$ from a distributed joint histogram $h(A, B)$. Let $c(a, b)$ be the frequency count of pair (a, b) of intensity value appeared in $h(A, B)$. Then, probability distributions $p_A(a)$ and $p_{AB}(a, b)$ required for $\mathcal{C}_{\text{similarity}}(A, B)$ can be computed by

$$p_A(a) = \frac{c(a)}{\sum_{a,b} c(a, b)}, \tag{9}$$

$$p_{AB}(a, b) = \frac{c(a, b)}{\sum_{a,b} c(a, b)}, \tag{10}$$

where $c(a) = \sum_b c(a, b)$. Substituting Eqs. (9) and (10) to Eqs. (3)–(5) gives

$$\mathcal{C}_{\text{similarity}}(A, B) = \frac{(-E_1 + E_4 \log E_4) + (-E_2 + E_4 \log E_4)}{-E_3 + E_4 \log E_4}, \tag{11}$$

where $E_1 = \sum_a c(a) \log c(a)$, $E_2 = \sum_b c(b) \log c(b)$, $E_3 = \sum_{a,b} c(a, b) \log c(a, b)$, and $E_4 = \sum_{a,b} c(a, b)$. Note here that $E_1, \ldots, E_4$ are represented as reduction operations over the joint histogram $h(A, B)$. Therefore, in order to compute $E_1, \ldots, E_4$, the algorithm sweeps $h(A, B)$ in the row and column directions, as illustrated in Fig. 5. This sweep operation can be parallelized by means of data-parallel processing followed by one collective communication as follows.

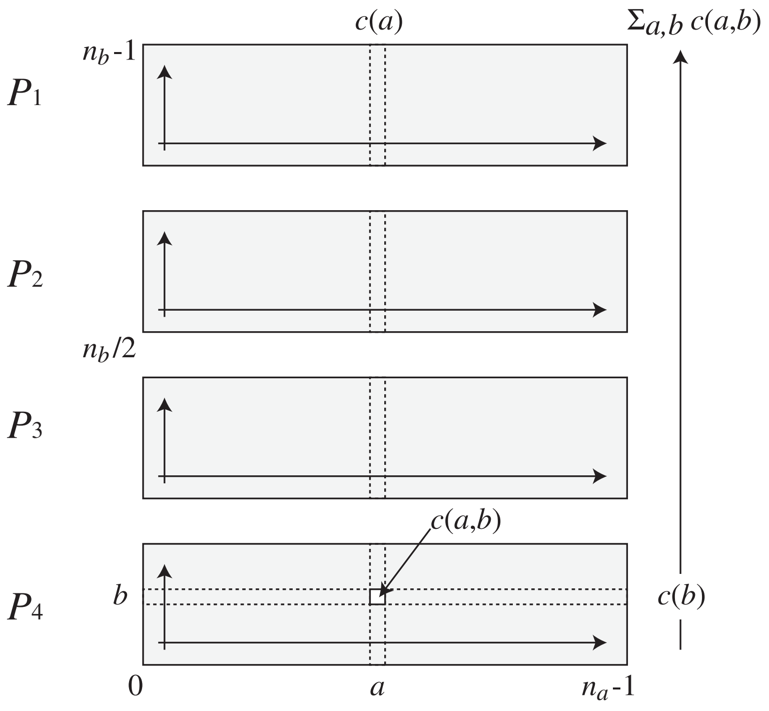


Fig. 5. Reduction operation for similarity computation on 4 processors. Given distributed $n_a \times n_b$ joint histogram $h(R, \mathbf{T}(F))$, each processor locally sweeps its responsible histogram in the row and column directions, and then participates in a collective communication in order to perform a global sweep in the column direction.

(c) Parallel sweep: For all $1 \le s \le N$, processor P_s independently sweeps its responsible portion of $h(A, B)$ in the row and column directions so that obtains local sums of $E_1, \ldots, E_4$.
(d) Similarity computation: All processors participate in a collective communication to reduce local sums into global sums $E_1, \ldots, E_4$. After this reduction operation, processor P_1 computes $\mathcal{C}_{\text{similarity}}(A, B)$ according to Eq. (11).
(d) Result distribution: P_1 broadcasts $\mathcal{C}_{\text{similarity}}(A, B)$ to all processors.

Note here that processors are allowed to communicate only reduced values instead of 2-D histograms, because the BS method generates $h(A, B)$ in a distributed manner. Thus, this distributed histogram generation reduces the amount of communication. Furthermore, because every processor generates an equally sized portion of $h(A, B)$, it allows processors to immediately begin the succeeding sweep operation with perfect load balancing.

Table 1
Parameter values used for experiments. L, δ, γ, d, and ϵ represent deformation level, control point spacing, image sampling rate, length for deformable region, and threshold for optimization, respectively.

L	δ	γ	d	ϵ	Image size	Data distribution
	(mm)	(mm)	(mm)		(voxel)	
1	42.88	2.68	14	0.001	$128 \times 128 \times 74$	Off
2	21.44	1.34	14	0.001	$256 \times 256 \times 148$	Off
3	10.72	0.67	14	0.001	$512 \times 512 \times 295$	On
4	5.36	0.34	14	0.001	$1024 \times 1024 \times 590$	On

5 Experimental Results

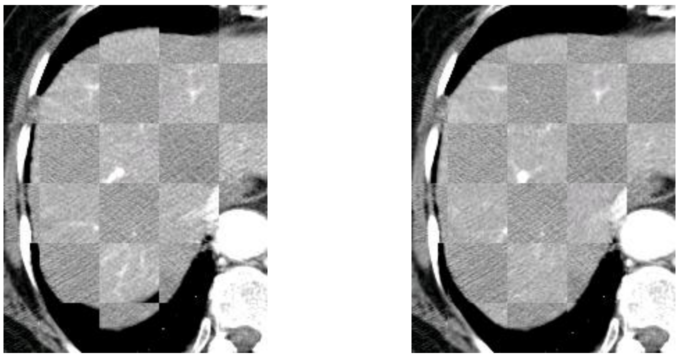
To evaluate our parallel algorithm on its execution time and memory usage, we have implemented it on a cluster of PCs. Our implementation uses the C++ language and MPICH-SCore library [36], a highly portable and efficient implementation of the Message Passing Interface (MPI) standard [21].

The cluster is composed of 64 nodes, each with two Pentium III 1-GHz CPUs and 2 GB of main memory. These nodes connect to Myrinet [37] and Fast Ethernet switches, which provide the link bandwidth of 2 Gb/s and that of 100 Mb/s, respectively. In addition to these computing nodes, there is a file server with a Gigabit Ethernet adapter that provides an NFS mounted storage for the nodes.

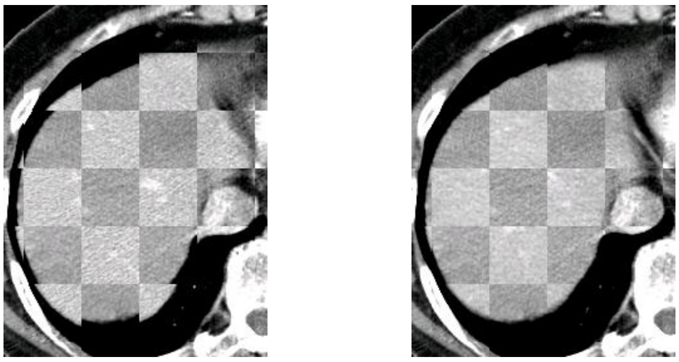
5.1 Registration Process

We applied our algorithm to four datasets of contrast-enhanced liver CT images, which have a size of $512 \times 512 \times 159$ voxels with spatial resolution of $0.67 \times 0.67 \times 1.25$ mm. Table 1 shows the parameter values used for the experiments. We refined control mesh Φ by four levels with reducing image sampling rate γ and control point spacing δ. As we mentioned in Section 4.2, this image resampling reduces the data size, so that the resampled images at the first and second levels were small enough to carry out on-memory computation without data distribution. Therefore, we distributed data only at the third and last levels.

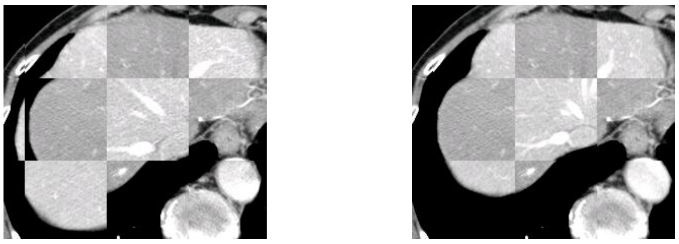
We initialized status S in accordance with $\alpha = 0.65$ at the beginning of each deformation level and updated it by $\alpha = 0.005$ after each deformation. In addition to these coefficient values for reference image and joint image pair measures, threshold ϵ for optimization is also experimentally determined.



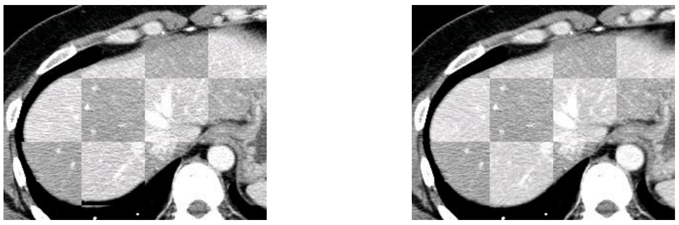
(a)



(b)



(c)



(d)

Fig. 6. Checkerboard visualization of four contrast-enhanced liver CT images: (a) dataset #1; (b) dataset #2; (c) dataset #3; and (d) dataset #4 obtained from different subjects. Reference and floating images in the left-hand side are aligned after registration, as presented in the right-hand side. Maximum deformations were 15.7, 11.4, 16.1, and 10.1 mm, respectively.

Table 2
Execution time measured on Myrinet and Fast Ethernet for four datasets. Time is presented in minutes. Registration tasks are stopped immediately after the third deformation level.

N	Myrinet (m)				Fast Ethernet (m)			
	#1	#2	#3	#4	#1	#2	#3	#4
1	1073.4	1093.5	1001.2	741.4	1073.4	1093.5	1001.2	741.4
8	139.2	141.9	127.8	96.6	142.7	144.6	133.0	98.5
16	71.2	71.5	66.0	49.2	75.7	78.3	71.5	53.7
32	37.2	37.4	35.1	26.3	43.2	43.7	44.1	32.8
64	19.8	20.0	19.1	14.3	27.7	27.4	27.6	21.3
128	11.2	11.2	10.1	7.8	20.2	20.5	20.4	16.6

Table 3
Speedup measured on Myrinet and Fast Ethernet for four datasets.

N	Myrinet				Fast Ethernet			
	#1	#2	#3	#4	#1	#2	#3	#4
8	7.7	7.7	7.8	7.8	7.5	7.6	7.5	7.5
16	15.0	15.3	15.2	15.1	14.2	14.0	14.0	13.8
32	28.8	29.2	28.5	28.2	24.8	25.0	22.7	22.6
64	54.3	54.7	52.3	51.9	38.8	39.8	36.3	34.7
128	95.9	97.6	99.2	95.1	53.1	53.2	49.0	44.7

Fig. 6 shows an example of registration results. In this figure, the reference and floating images are alternately shown in a checkerboard pattern. For all datasets, the maximum deformations were between 11.4 and 16.1 mm. Therefore, a marginal region with length $d = 14$ mm is large enough for these datasets, because it allows deformations with less than $14\sqrt{2}$ mm.

5.2 Execution Time

Tables 2 and 3 show the execution time and speedup measured for four datasets. During this measurement, we stopped registration tasks immediately after the third level, because the finest level takes more than a half hour (as presented later in Section 5.3), which is unacceptable to clinical use.

On 128 processors with Myrinet, our algorithm reduces the execution time approximately from 15 hours to 10 minutes with high speedups ranging from a factor of 95 to that of 99. Thus, our algorithm realizes large-scale registration with a scal-

Table 4
Breakdown of execution time for dataset #1. Time is presented in minutes.

N	Execution time on Myrinet (m)				Execution time on Fast Ethernet (m)			
	Gradient	Similarity	Others	Total	Gradient	Similarity	Others	Total
1	991.7	73.6	8.1	1073.4	991.7	73.6	8.1	1073.4
8	127.2	10.2	1.8	139.2	129.4	10.8	2.5	142.7
16	64.9	5.1	1.2	71.2	67.8	5.8	2.1	75.7
32	33.6	2.7	0.9	37.2	37.3	3.8	2.1	43.2
64	17.6	1.4	0.8	19.8	22.8	2.5	2.4	27.7
128	9.6	0.8	0.8	11.2	15.4	2.6	2.2	20.2

Table 5
Ratio of communication time for dataset #1. Communication time contains data transfer time and waiting time.

N	Ratio on Myrinet (%)				Ratio on Fast Ethernet (%)			
	Gradient	Similarity	Others	Total	Gradient	Similarity	Others	Total
8	0.07	0.07	0.83	0.08	1.8	5.6	28.6	2.5
16	0.11	0.15	1.63	0.13	4.4	12.2	43.8	6.1
32	0.13	0.29	2.44	0.19	10.0	29.2	58.2	14.1
64	0.20	0.55	3.53	0.36	23.0	44.3	67.8	28.8
128	0.54	0.82	5.49	0.83	38.0	69.5	65.6	45.0

able speedup on Myrinet. In contrast, it also offers high speedup on Fast Ethernet when using less than 32 processors. However, it decreases parallel efficiency as the number of processors increases, so that Myrinet provides twice shorter execution time compared to Fast Ethernet on 128 processors. This performance loss is due to the lack of the link bandwidth, as presented later in this section.

Table 4 shows the breakdown of the execution time for dataset #1. On 128 processors with Myrinet, our algorithm accelerates the gradient computation and the similarity computation by a factor of 103 and that of 92, respectively. Both these high speedups contribute to the total speedup of a factor of 96. If the algorithm performs the similarity computation in sequential, the execution time on 128 processors can be estimated as 84 minutes, resulting in a low speedup of a factor of 13, as we mentioned in Section 4.1.

The remaining time corresponds to other computations, such as image loading and resampling. In our current implementation, we experimentally tuned input/output (I/O) performance to our cluster, so that 4 nodes simultaneously load a quarter of images from the NFS server through Fast Ethernet, and then broadcast it to the remaining nodes in order to distribute the responsible portion of data. In spite of this tuning, we obtained a low speedup of a factor of 9 for image loading and resampling. This low speedup is mainly due to disk access that takes about 25 seconds to load images F and R from the NFS server. Therefore, file I/O can reveal as a performance bottleneck on large-scale clusters if we are unable to preload the images before surgery, for example, in a case where intraoperatively obtained images are essential to guide surgeons during surgery.

Table 5 shows the ratio of the communication time for each part presented in Table 4. Here, the communication time is defined as the sum of data transfer time and waiting time. We can see that our algorithm running on Myrinet realizes a low ratio of at most 0.83%. Thus, most of the execution time on Myrinet is spent in computation. In contrast, on Fast Ethernet, the ratio increases with the number of processors and reaches 45.0% when using 128 processors. Therefore, the communication time must be further reduced for this low-bandwidth, high-latency network.

One solution for reducing the communication time is to improve the compositing method in the similarity computation by exploiting the sparsity of the transmitting data. For example, some BS-based methods [38, 39] integrate a lossless data compression algorithm into the BS method in order to realize this reduction. Other methods [40, 41] also achieve this by using categorized pixel information, which aims at minimizing both the amount and the occurrence of communication.

To make clear the benefits of our load balancing technique, we now compare the following three algorithms.

- PDL: Parallel nonrigid registration with data distribution and load balancing.
- PD: Parallel nonrigid registration only with data distribution.
- PL: Parallel nonrigid registration only with load balancing.

Fig. 7 shows the speedup for the optimization at the third level. On 128 processors with Myrinet, our load balancing algorithm increases the speedup by three times, from a factor of 33 (PD) to that of 103 (PDL). However, the speedup of PDL is 16% lower than that of PL, a factor of 123. Thus, our algorithm enables data distribution with low performance loss.

Fig. 8 shows the distribution of the execution time required for an iteration of the gradient computation on Myrinet. In PD, processor P_7 takes responsibility for the maximum of 185 active control points, so that it takes the maximum of 436.3 seconds for the gradient computation while several processors take less than 0.1 seconds. In contrast, PDL balances the processor workloads, so that the maximum time reduces to 94.7 seconds. For example, our load balancing technique enables high-loaded processor P_7 to distribute its workload to six low-loaded processors P_1, P_{29}, P_{35}, P_{49}, P_{53}, and P_{57}. Therefore, our PDL achieves better load balance compared to PD, which statically assigns equally sized domains to processors. Actually, the standard deviations of the execution time in PDL, PD, and PL are 9.28, 107.4,

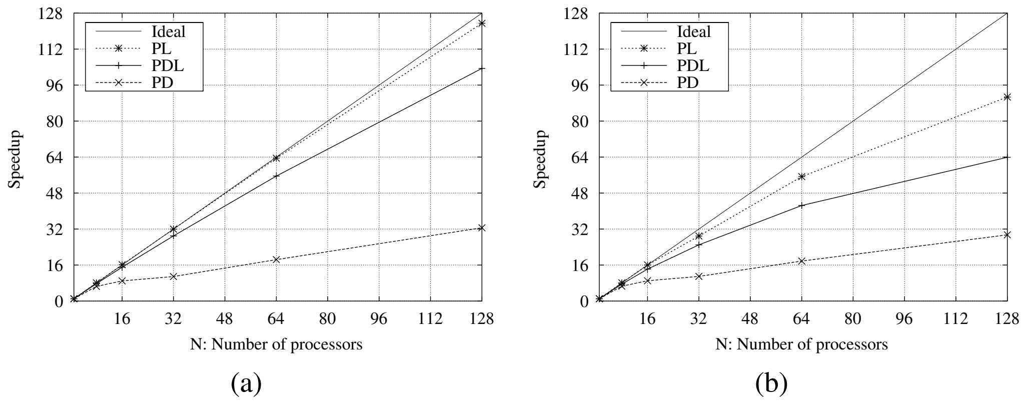


Fig. 7. Speedup measured on (a) Myrinet and (b) Fast Ethernet for optimization at the third deformation level.

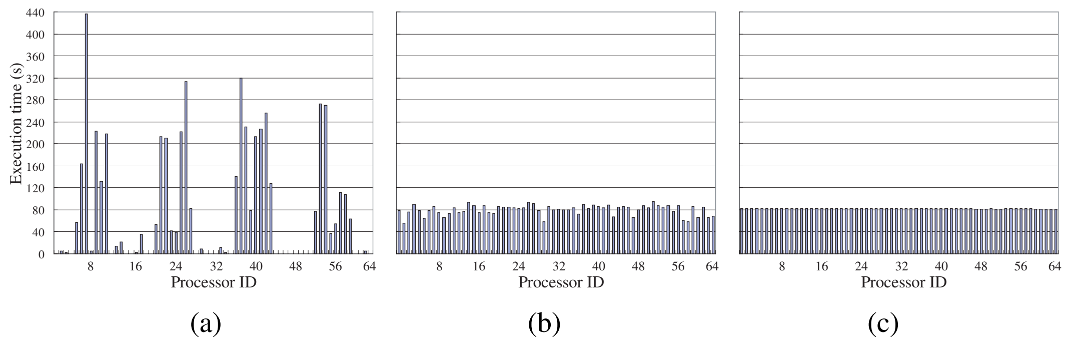


Fig. 8. Distribution of execution time required for an iteration of gradient computation in (a) PD, (b) PDL, and (c) PL, running on Myrinet.

and 0.37 seconds on Myrinet and 9.29, 108.3, and 0.44 seconds on Fast Ethernet, respectively.

Although PDL requires data redistribution from high-loaded to low-loaded processors, it takes only 1.1 seconds on 128 processors with Myrinet. For all four datasets, the amount of transmitted data is 28.8 MB composed mainly of subimages $F_{t,1}$ and $R_{t,1}$ with many active control points $\Phi^{+}_{t,1}$, where $1 \leq t \leq M$. Furthermore, although $|G_t|$ ranges from 2 to 7 processors, the communication imbalance associated with this redistribution has little effect on the total performance on Myrinet. However, it is a performance bottleneck on Fast Ethernet due to low bandwidth. Thus, although load balancing is achieved on Fast Ethernet, data redistribution takes long time on this low bandwidth network, and thereby Fast Ethernet takes twice longer time compared to Myrinet.

Table 6
Breakdown of memory usage in MB. PL fails to perform registration tasks at the finest level due to the lack of physical memory.

L	Algorithm	Breakdown				Total
		u_1: Image	u_2: Margin	u_3: Shared	Others	
3	PL	295.0	—	—	54.2	349.2
	PD	4.6	24.2	—	29.1	57.9
	PDL	4.6	24.2	28.8	20.8	78.4
4	PL	2355.0	—	—	Unknown	At least 2355.0
	PD	37.0	97.4	—	113.4	247.8
	PDL	37.0	97.4	134.4	59.3	328.1

5.3 Memory Usage

Table 6 shows the breakdown of memory usage required for each algorithm. In this table, u_1, u_2, and u_3 represent the data size for subimages without marginal regions, that for marginal regions, and that for shared subimages redistributed to low-loaded processors, respectively. The total amount indicates that our PDL reduces the amount of memory usage by 77% and 86% compared to PL at the third and finest levels, respectively. It also shows that PDL needs approximately 35% additional amount of memory compared to PD.

Note here that $(u_1 + u_2)/u_1$ approaches to 1.0 as deformation level L increases, where $u_1 + u_2$ denotes the amount of memory usage for subimages with marginal regions. For example, $(u_1 + u_2)/u_1$ indicates that partially overlapped images require approximately six times more memory resources than disjoint images at the third level, but three times at the finest level. This decrease can be explained as follows. Increasing L makes δ smaller, so that l $(= d + 2\delta)$ decreases as L increases. Therefore, as the resolution of images increases, the memory usage for PDL approaches to that for PD.

As shown in Table 6, data distribution technique enables us to register images at the finest resolution level. Without data distribution, the optimization at this level requires more than 2 GB of physical memory to load the entire of two $1024 \times 1024 \times 590$ voxel images, so that out-of-core computation occurs on our cluster. Since this computation takes significant execution time, PL is unsuitable for high-resolution image registration. On the other hand, PD enables on-memory computation, however, it takes 105 minutes on 128 processors while PDL takes 47 minutes on Myrinet. Thus, our PDL algorithm increases the resolution of images available on a system with high efficiency provided by a load balancing capability.

Fig. 9 shows the optimization progress on 128 processors with Myrinet. It is clear

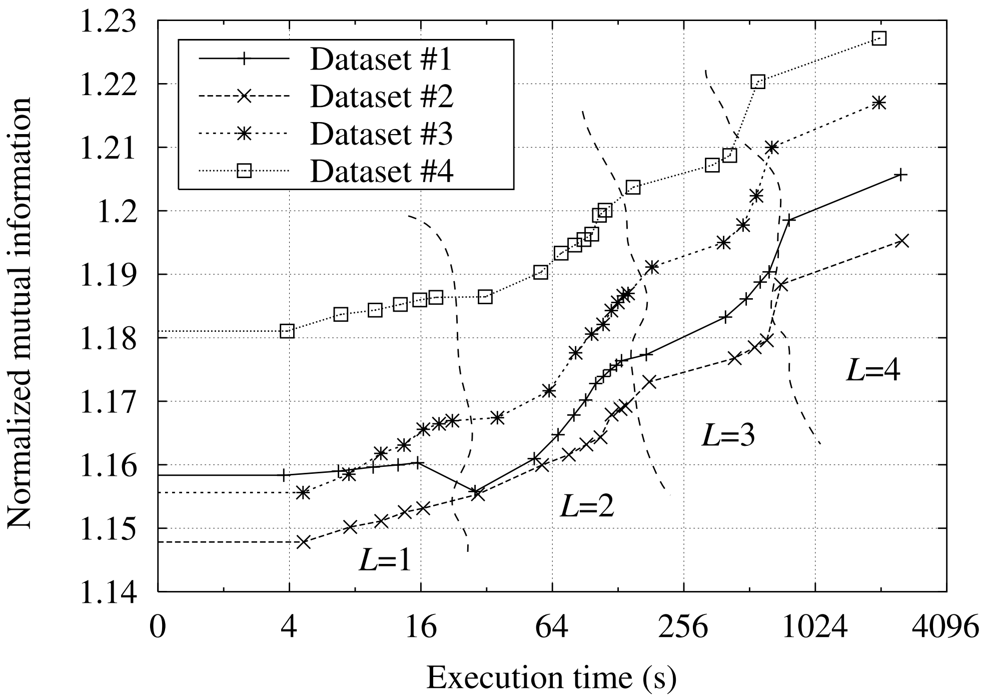


Fig. 9. Optimization progress on 128 processors using Myrinet.

that increasing deformation levels obtain more accurate results. For dataset #3, the optimization at the finest level raises NMI by 0.015 while the total improvement is 0.067. Since this level takes much longer time than the former levels to complete registration, deciding the finest level in which the registration is terminated is important for clinical use, where a series of images can be produced in rapid succession.

5.4 Discussion

The above experimental results show that our algorithm running on a 128-node cluster with Myrinet interconnection satisfies the time constraints for the following two scenarios.

- Intraoperative registration, which requires highly available, dedicated HPC resources to perform critical missions in a rapid, reliable manner. Our algorithm running on a dedicated cluster reduces the registration time to a clinically compatible time of approximately 10 minutes, so that it realizes intraoperative registration for datasets up to $512 \times 512 \times 295$ voxel images.
- Preoperative registration, which requires HPC resources to perform medical diagnosis for developing an accurate, detailed surgical plan based on high-resolution images. Our algorithm aligns $1024 \times 1024 \times 590$ voxel images in about 50 minutes. Though this acceleration is not sufficient for surgical use, it is short enough for this diagnosis. As compared to intraoperative procedures, such preoperative

procedures are not so severe on the time constraints, because their objective is to plan surgical strategy for obtaining better surgical outcomes with increased surgical safety, well in advance of the surgery date. For example, high-resolution image registration is necessary for accurate detection of cancers, which minimizes the removal of normal tissue.

In the following, we discuss the usability of the Grid infrastructure for the above two scenarios. To use the Grid infrastructure for these scenarios, we must consider the two differences between clusters and Grids, as follows.

- Decentralized control of resources. The first concern is the decentralized control of Grid resources [24], because the Grid integrates and coordinates resources from multiple organizations. This decentralized nature of the Grid causes difficulties in using itself for intraoperative registration. To perform this critical mission in a rapid, reliable manner, Grid resource brokers must monitor the status of resources to provide highly available, dedicated resources in a short time. From the viewpoint of giving a realistic solution to these qualitative requirements, dedicated clusters with centralized control systems are more stable and secure than Grids, because such centralized system can easily provide sufficient resources whenever a registration task is submitted, and guarantee that the allocated resources will be dedicated to the task until its completion. This is not easy for the Grid, where resources such as processors, memory, and interconnection belong to different control domains.
- Lower performance of interconnection. The second concern is the Grid interconnection, which usually has lower bandwidth and higher latency than the cluster interconnection. Due to this slower network, Grid applications must minimize the amount of communication to obtain higher performance. As we mentioned in Section 5.2, lossless data compression algorithms will play an important role in achieving this minimization if the link bandwidth is less than 100 Mb/s. In addition to this application layer issue, resource brokers also must rapidly gather the status of resources and provide them through this slower network. If a Grid consists of less than a few hundred nodes, this resource management will not be a significant problem in recent brokers such as the Globus Toolkit Monitoring and Discovery Service (MDS) [42] and the Condor Hawkeye [43], because Zhang's performance study [44] shows that they take less than 10 seconds to gather the status from 200 nodes. However, this study also shows that the Condor Hawkeye takes about one minute on 1000 nodes, so that as the number of nodes increases, resource selection will become a performance bottleneck when using the Grid for intraoperative registration.

Summarizing the above discussion, we think that it is not easy to use the Grid infrastructure for surgical use mainly due to its decentralized nature rather than the time constraints. However, this infrastructure is attractive for medical diagnosis, because such preoperative procedures are not critical missions, and combining our algorithm with a lossless data compression algorithm will satisfy the timing constraints for this use.

6 Conclusions and Future Work

We have presented a data distributed parallel algorithm for nonrigid image registration on distributed memory multiprocessors. To realize large-scale registration with scalable performance, our algorithm uses a robust and efficient registration algorithm [11] with three techniques: data distribution, data-parallel processing, and load balancing.

The experimental results show that our data distributed algorithm reduces the amount of memory usage by at least 77% compared to a data undistributed algorithm, enabling us to increase the resolution of images available on a system. Furthermore, with 35% additional amount of memory, our load balancing technique reduces the execution time in half. As a result, our algorithm enables nonrigid registration of liver CT images of $512 \times 512 \times 295$ voxels in less than 10 minutes on 128 processors, which takes approximately 15 hours on a single CPU system.

Future work includes the development of a load balancing technique and the integration of a lossless data compression algorithm that further improves the performance on low-bandwidth, high-latency network with many nodes, for example, the Grid.

Acknowledgments

The authors would like to thank Dr. Jun Masumoto of Medical Imaging Laboratory, Ltd., Dr. Yoshinobu Sato and Dr. Shinichi Tamura of the Graduate School of Medicine, Osaka University for useful discussions regarding this work. We would like to thank the anonymous reviewers for their valuable comments.

References

[1] J. V. Hajnal, D. L. Hill, D. J. Hawkes (Eds.), Medical Image Registration, CRC Press, Boca Raton, FL, 2001.

[2] J. P. W. Pluim, J. B. A. Maintz, M. A. Viergever, Mutual-information-based registration of medical images: A survey, IEEE Trans. Medical Imaging 22 (8) (2003) 986–1004.

[3] A. Guéziec, P. Kazanzides, B. Williamson, R. H. Taylor, Anatomy-based registration of CT-scan and intraoperative X-ray images for guiding a surgical robot, IEEE Trans. Medical Imaging 17 (5) (1998) 715–728.

[4] S. K. Warfield, M. Ferrant, X. Gallez, A. Nabavi, F. A. Jolesz, R. Kikinis, Real-time biomechanical simulation of volumetric brain deformation for image guided neurosurgery, in: Proc. High Performance Networking and Computing Conf. (SC2000), 2000, pp. 1–16.

[5] T. Rohlfing, C. R. Maurer, Nonrigid image registration in shared-memory multiprocessor environments with application to brains, breasts, and bees, IEEE Trans. Information Technology in Biomedicine 7 (1) (2003) 16–25.

[6] U. Pietrzyk, K. Herholz, A. Schuster, H.-M. V. Stockhausen, H. Lucht, W.-D. Heiss, Clinical applications of registration and fusion of multimodality brain images from PET, SPECT, CT, and MRI, European J. Radiology 21 (1996) 174–182.

[7] F. Maes, A. Collignon, D. Vandermeulen, G. Marchal, P. Suetens, Multimodality image registration by maximization of mutual information, IEEE Trans. Medical Imaging 16 (2) (1997) 187–198.

[8] J. G. Rosenman, E. P. Miller, G. Tracton, T. J. Cullip, Image registration: an essential part of radiation therapy treatment planning, Int'l J. Radiation Oncology, Biology, Physics 40 (1) (1998) 197–205.

[9] D. Rueckert, L. I. Sonoda, C. Hayes, D. L. G. Hill, M. O. Leach, D. J. Hawkes, Nonrigid registration using free-form deformations: Application to breast MR images, IEEE Trans. Medical Imaging 18 (8) (1999) 712–721.

[10] A. Carrillo, J. L. Duerk, J. S. Lewin, D. L. Wilson, Semiautomatic 3-D image registration as applied to interventional MRI liver cancer treatment, IEEE Trans. Medical Imaging 19 (3) (2000) 175–185.

[11] J. A. Schnabel, D. Rueckert, M. Quist, J. M. Blackall, A. D. Castellano-Smith, T. Hartkens, G. P. Penney, W. A. Hall, H. Liu, C. L. Truwit, F. A. Gerritsen, D. L. G. Hill, D. J. Hawkes, A generic framework for non-rigid registration based on non-uniform multi-level free-form deformations, in: Proc. 4th Int'l Conf. Medical Image Computing and Computer-Assisted Intervention (MICCAI'01), Lecture Notes in Computer Science, Vol. 2208, 2001, pp. 573–581.

[12] T. Rohlfing, C. R. Maurer, D. A. Bluemke, M. A. Jacobs, Volume-preserving nonrigid registration of MR breast images using free-form deformation with an incompressibility constraint, IEEE Trans. Medical Imaging 22 (6) (2003) 730–741.

[13] G. E. Christensen, MIMD vs. SIMD parallel processing: A case study in 3D medical image registration, Parallel Computing 24 (9/10) (1998) 1369–1383.

[14] S. Warfield, F. Jolesz, R. Kikinis, A high performance computing approach to the registration of medical imaging data, Parallel Computing 24 (9/10) (1998) 1345–1368.

[15] S. Ourselin, R. Stefanescu, X. Pennec, Robust registration of multi-modal images: Towards real-time clinical applications, in: Proc. 5th Int'l Conf. Medical Image

Computing and Computer-Assisted Intervention (MICCAI'02), Part II, Lecture Notes in Computer Science, Vol. 2489, 2002, pp. 140–147.

[16] T. Butz, J.-P. Thiran, Affine registration with feature space mutual information, in: Proc. 4th Int'l Conf. Medical Image Computing and Computer-Assisted Intervention (MICCAI'01), Lecture Notes in Computer Science, Vol. 2208, 2001, pp. 549–556.

[17] S. Hastings, T. Kurc, S. Langella, U. Catalyurek, T. Pan, J. Saltz, Image processing for the grid: A toolkit for building grid-enabled image processing applications, in: Proc. 3rd IEEE/ACM Int'l Symp. Cluster Computing and the Grid (CCGrid'03), 2003, pp. 36–43.

[18] S. Kleiman, D. Shah, B. Smaalders (Eds.), Programming with Threads, Prentice Hall PTR, Englewood Cliffs, NJ, 1995.

[19] T. E. Anderson, D. E. Culler, D. A. Patterson, the NOW Team, A case for NOW (networks of workstations), IEEE Micro 15 (1) (1995) 54–64.

[20] T. L. Sterling, D. Savarese, D. J. Becker, J. E. Dorband, U. A. Ranawake, C. V. Packer, BEOWULF: A parallel workstation for scientific computation, in: Proc. 24th Int'l Conf. Parallel Processing (ICPP'95), 1995, pp. 11–14.

[21] Message Passing Interface Forum, MPI: A message-passing interface standard, Int'l J. Supercomputer Applications and High Performance Computing 8 (3/4) (1994) 159–416.

[22] R. Chandra, L. Dagum, D. Kohr, D. Maydan, J. McDonald, R. Menon, Parallel Programming in OpenMP, Morgan Kaufmann Publishers, San Mateo, CA, 2000.

[23] I. Foster, C. Kesselman (Eds.), The Grid: Blueprint of a New Computing Infrastructure, Morgan Kaufmann Publishers, San Mateo, CA, 1998.

[24] I. Foster, What is the Grid? a three point checklist, 2002, available at http://www-fp.mcs.anl.gov/ foster/Articles/WhatIsTheGrid.pdf.

[25] I. Foster, C. Kesselman, Globus: A metacomputing infrastructure toolkit, Int'l J. Supercomputer Applications and High Performance Computing 11 (2) (1997) 115–128.

[26] S. Lee, G. Wolberg, S. Y. Shin, Scattered data interpolation with multilevel B-splines, IEEE Trans. Visualization and Computer Graphics 3 (3) (1997) 228–244.

[27] C. Studholme, D. L. G. Hill, D. J. Hawkes, An overlap invariant entropy measure of 3D medical image alignment, Pattern Recognition 32 (1) (1999) 71–86.

[28] F. L. Bookstein, Principal warps: thin-plate splines and the decomposition of deformations, IEEE Trans. Pattern Analysis and Machine Intelligence 11 (6) (1989) 567–585.

[29] M. H. Davis, A. Khotanzad, D. P. Flamig, S. E. Harms, A physics-based coordinate transformation for 3-D image matching, IEEE Trans. Medical Imaging 16 (3) (1997) 317–328.

[30] T. Rohlfing, C. R. Maurer, Intensity-based non-rigid registration using adaptive multilevel free-form deformation with an incompressibility constraint, in: Proc. 4th Int'l Conf. Medical Image Computing and Computer-Assisted Intervention (MICCAI'01), Lecture Notes in Computer Science, Vol. 2208, 2001, pp. 111–119.

[31] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery, NUMERICAL RECIPES in C: The Art of Scientific Computing, Cambridge University Press, Cambridge, UK, 1988.

[32] G. K. Rohde, A. Aldroubi, B. M. Dawant, Adaptive free-form deformation for inter-patient medical image registration, in: Proc. SPIE Medical Imaging: Image Processing, 2001, pp. 1578–1587.

[33] G. Amdahl, Validity of the single processor approach to achieving large-scale computing capabilities, in: Proc. AFIPS Conf., Vol. 30, 1967, pp. 483–485.

[34] R. L. Graham, Bounds on multiprocessing timing anomalies, SIAM J. Applied Mathematics 17 (1969) 416–429.

[35] K.-L. Ma, J. S. Painter, C. D. Hansen, M. F. Krogh, Parallel volume rendering using binary-swap compositing, IEEE Computer Graphics and Applications 14 (4) (1994) 59–68.

[36] F. O'Carroll, H. Tezuka, A. Hori, Y. Ishikawa, The design and implementation of zero copy MPI using commodity hardware with a high performance network, in: Proc. 12th ACM Int'l Conf. Supercomputing (ICS'98), 1998, pp. 243–250.

[37] N. J. Boden, D. Cohen, R. E. Felderman, A. E. Kulawik, C. L. Seitz, J. N. Seizovic, W.-K. Su, Myrinet: A gigabit-per-second local-area network, IEEE Micro 15 (1) (1995) 29–36.

[38] A. Takeuchi, F. Ino, K. Hagihara, An improved binary-swap compositing for sort-last parallel rendering on distributed memory multiprocessors, Parallel Computing 29 (11/12) (2003) 1745–1762.

[39] K. Sano, Y. Kobayashi, T. Nakamura, Differential coding scheme for efficient parallel image composition on a PC cluster system, Parallel Computing 30 (2) (2004) 285–299.

[40] F. Ino, T. Sasaki, A. Takeuchi, K. Hagihara, A divided-screenwise hierarchical compositing for sort-last parallel volume rendering, in: Proc. 17th Int'l Parallel and Distributed Processing Symposium (IPDPS'03), 2003.

[41] A. Stompel, K.-L. Ma, E. B. Lum, J. Ahrens, J. Patchett, SLIC: Scheduled linear image compositing for parallel volume rendering, in: Proc. 2nd IEEE Symp. Parallel and Large-Data Visualization and Graphics (PVG'03), 2003, pp. 33–40.

[42] K. Czajkowski, S. Fitzgerald, I. Foster, C. Kesselman, Grid information services for distributed resource sharing, in: Proc. 10th IEEE Int'l Symp. High Performance Distributed Computing (HPDC'01), 2001, pp. 181–194.

[43] Hawkeye, A monitoring and management tool for distributed systems, available at http://www.cs.wisc.edu/condor/hawkeye/.

[44] X. Zhang, J. L. Freschl, J. M. Schopf, A performance study of monitoring and information services for distributed systems, in: Proc. 12th IEEE Int'l Symp. High Performance Distributed Computing (HPDC'03), 2003, pp. 270–282.