SIIM-TR-A-**02-08**

SIIM Technical Report

Non-Linear Image Registration on PC-Clusters Using Parallel FFT Techniques

by

Martin Böhme, Rainer Hagenau,
Jan Modersitzki, Bodo Siebert

Schriftenreihe der Institute für
Informatik/Mathematik
Serie A
June 21, 2002

Universität zu Lübeck
Technisch-Naturwissenschaftliche Fakultät

Email: siebert@informatik.mu-luebeck.de
Phone: +49-451-7030-414
Fax: +49-451-7030-438

Non-Linear Image Registration on PC-Clusters Using Parallel FFT Techniques

Martin Böhme[*], Rainer Hagenau[†], Jan Modersitzki[‡], Bodo Siebert[§]

June 21, 2002

Abstract

Reliable image registration is a key problem within the three-dimensional reconstruction of data obtained by two-dimensional image stacks. Here, a parallel implementation of the so-called *elastic matching algorithm* approach is presented. This algorithm is based on a fixpointtype iteration, where in each step a linear system of equations has to be solved. To compute the solution of the linear systems fast fourier techniques are used. The algorithm as well as some numerical examples are presented.

1 Introduction

Image registration is a basic problem within medical image processing. Especially, if images arise from a series of sections through a part of the human body, e.g. CT (computer tomography), MRI (magnetic resonance imaging), or PET (positron emission tomography).

In our application, the images are high resolution flat bed scans of histological sections originating from a male human brain. The sectioning process is necessary in order to visualize finer structures of the brain, e.g. neurons and their locations. A specially adapted light microscope is used for further visualization purposes.

Fig. 1 displays the arbitrarily chosen sections S3799 (Fig. 1a) and S3800 (Fig. 1b) of a total of about 6000 frontal sections. The sectioning process destroys the three-dimensional structure of the tissue. A tissue particle might be displayed at two

[*]Medical University of Lübeck, Ratzeburger Allee 160, D-23538 Lübeck, Germany. E-mail: boehme@informatik.mu-luebeck.de

[†]Institute of Computer Engineering, Medical University of Lübeck, Ratzeburger Allee 160, D-23538 Lübeck, Germany. E-mail: hagenau@iti.mu-luebeck.de

[‡]Institute of Mathematics, Medical University of Lübeck, Wallstraße 40, D-23560 Lübeck, Germany. E-mail: modersitzki@math.mu-luebeck.de

[§]Institute for Theoretical Computer Science, Medical University of Lübeck, Wallstraße 40, D-23560 Lübeck, Germany. E-mail: siebert@tcs.mu-luebeck.de

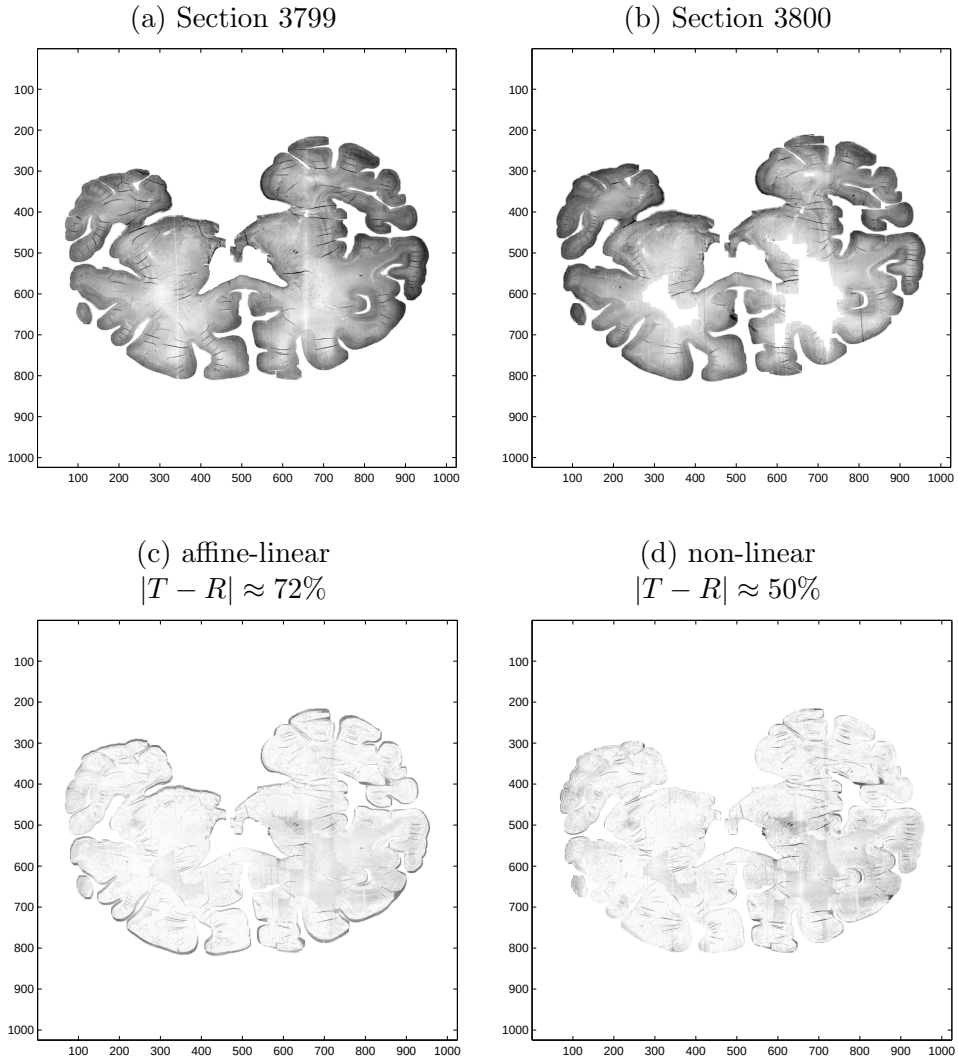

Figure 1: Arbitrarily chosen sections S3799 (a) and S3800 (b); difference images after of linear (c) and after non-linear (d) registration.

different geometrical positions in the images S3799 and S3800. Thus, registration techniques are required to recover the three-dimensional shape of the brain. Fig. 1 also shows the differences between the two registered images. For Fig. 1c we used an optimal affine-linear registration, cf. e.g. [15], and for Fig. 1d we used a non-linear registration based on an elasticity model.

A linear registration already reduces the geometrical distortion of the images considerably. For the above example, the difference after linear registration is about 72% of the one obtained by a human expert registration (100%). However, the reduction is not sufficient for this application. Hence, an additional non-linear registration was applied. Here the difference is about 50% of the expert registration and almost all structural differences in the images have been corrected, cf. Fig. 1d.

For the non-linear registration we used the so-called *elastic matching algorithm*.

This method is based on a linear elasticity model and is also used in other projects, e.g. [1], [2], [5], [6], [18].

We present a parallel implementation of a FFT-based algorithm for the elastic matching. Using an appropriate communication strategy and a high speed network on a PC-Cluster, it is possible to end up with almost linear speed-up.

Thus, the designed algorithm provides a basis for the registration of large images in a reasonable time.

The paper is organized as follows. A brief description of the underlying mathematics is given in Section 2. Section 3 presents our parallel implementation. Some numerical examples and the performance measurements are given in Section 4.

2 Mathematical background

Given two images $T, R : \Omega^2 \to \mathbb{R}$, where $\Omega := [0,1]^2$, we are looking for an elastic deformation $U : \mathbb{R}^2 \to \mathbb{R}^2$, $U(x,y) = (u(x,y), v(x,y))$, that simultaneously minimizes the difference between the deformed template and the reference image R, which serves as a model for the non-deformed template,

$$\begin{aligned} &\|T \circ U - R\|^2 \\ &= \int_{\Omega^2} [T(u(x,y), v(x,y)) - R(x,y)]^2 \; d(x,y) \end{aligned}$$

and the deformation energy

$$\begin{aligned} E(U) \;=\; &\int_\Omega \frac{\lambda}{2}\big(u_x + v_y\big)^2 \\ &\qquad +\mu\Big(u_x^2 + v_y^2 + \frac{1}{2}(u_y + v_x)^2\Big)\; d(x,y), \end{aligned}$$

where μ, λ are the so-called Lamé-constants, see e.g. [12]. This approach enforces similarity of the two images as well as connectivity of the tissue.

Applying the calculus of Euler-Lagrange, a minimizer is characterized by the so-called two-dimensional Navier-Lamé-equations, cf. e.g. [12],

$$\mathcal{A}U := \begin{pmatrix} \mu(u_{xx} + u_{yy}) + (\lambda + \mu)(u_{xx} + v_{xy}) \\ \mu(v_{xx} + v_{yy}) + (\lambda + \mu)(u_{xy} + v_{yy}) \end{pmatrix} = F(U). \tag{1}$$

Note,

$$F(U) = \big(T(x-u, y-v) - R(x,y)\big) \cdot \nabla T\big(x-u, y-v\big) \tag{2}$$

(which might be viewed as a force field) depends non-linearly on the deformation $U = (u, v)$.

An appropriate discretization of this equation finally leads to the following fixpointtype equation for the unknown discrete deformation field,

$$A \cdot U^{(k+1)} = F(U^{(k)}), \tag{3}$$

where $U^{(k)} = \left(u_{i,j}^{(k)}, v_{i,j}^{(k)}\right)$ and $F^{(k)} = F(U^{(k)}) = \left(f_{i,j}^{(k)}, g_{i,j}^{(k)}\right)$ are the discretized values of the deformation and force fields, respectively. Note, $F^{(k)}$ depends on $U^{(k)}$. An algorithm is summarized in Tab. 1, for details we refer to [8].

The main computational work in each iteration is the solution of the linear system (3). In [8] it has been shown that the matrix A can be factorized as $A = FDF^{-1}$, where F is essentially a Fourier-matrix and D is a 2×2 block-matrix with diagonal blocks D^p, $p = 1, 2, 3, 4$. Thus, the linear systems can be solved efficiently using FFT techniques. The implementation details for this process are listed in Tab. 2, the derivation is given in [8].

For the numerical examples presented in this paper we used the five-point stencils given in [8, Eq. (6) and (7)]. With this particular choices and an $m \times n$ discretization we have, cf. [8, Theorem 4],

$$\begin{aligned}
D_{i,j}^1 &:= -2(3\mu+\lambda) + 2(2\mu+\lambda)\cos(\tfrac{2\pi i}{m-1}) \\
&\quad +2\mu\cos(\tfrac{2\pi j}{n-1}), \\
D_{i,j}^4 &:= -2(3\mu+\lambda) + 2(2\mu+\lambda)\cos(\tfrac{2\pi j}{n-1}) \\
&\quad +2\mu\cos(\tfrac{2\pi j}{m-1}), \\
D_{i,j}^2 &:= D_{i,j}^3 := -4(\mu+\lambda)\sin(\tfrac{2\pi i}{m-1})\sin(\tfrac{2\pi j}{n-1}), \\
&\begin{pmatrix} D_{j,k}^{1,\dagger} & D_{j,k}^{2,\dagger} \\ D_{j,k}^{2,\dagger} & D_{j,k}^{4,\dagger} \end{pmatrix} := \begin{pmatrix} D_{j,k}^1 & D_{j,k}^2 \\ D_{j,k}^2 & D_{j,k}^4 \end{pmatrix}^\dagger,
\end{aligned}$$

where † denotes the Moore-Penrose pseudoinverse, cf. [10], and the numbers $D_{j,k}^{p,\dagger}$ are used in the algorithm, cf. Tab. 2.

3 Parallel implementation

Although the techniques proposed in [8] provide an efficient tool for solving the linear systems within the elastic matching algorithm, the problem remains challenging for today's computers when applied to three-dimensional registration and/or high resolution images:

1. the number of unknowns is $2mn$, e.g. 2.097.152 unknowns when matching two 1024×1024 images;
2. due to model restrictions the number of iterations needed might become large, e.g. $k_{\max} \approx 100$ in our application;

Table 1: Elastic-matching algorithm.

1. Choose initial discrete deformation, e.g. $U^{(0)} = 0$.
2. For $k = 0, 1, 2, \ldots, k_{\max}$,
   compute actual forces $F^{(k)} = F(U^{(k)})$, cf. Eq. 2,
   solve the linear system $A \cdot U^{(k+1)} = F^{(k)}$.

Table 2: Algorithm for solving the linear system (3) (based on 2D-FFT).

Compute $(\widetilde{f}_{i,j}) = \texttt{fft2}(f_{i,j})$, $(\widetilde{g}_{i,j}) = \texttt{fft2}(g_{i,j})$.
For $j = 2, \ldots, n-1$, for $i = 2, \ldots, m-1$ do
$\widetilde{u}_{i,j} = D^{1,\dagger}_{i,j}\widetilde{f}_{i,j} + D^{2,\dagger}_{i,j}\widetilde{g}_{i,j}$,
$\widetilde{v}_{i,j} = D^{3,\dagger}_{i,j}\widetilde{f}_{i,j} + D^{4,\dagger}_{i,j}\widetilde{g}_{i,j}$.
End.
Compute $(u_{i,j}) = \texttt{fft2}^{-1}(\widetilde{u}_{i,j})$, $(v_{i,j}) = \texttt{fft2}^{-1}(\widetilde{v}_{i,j})$.

3. the whole stack of images has to be matched, e.g about 6000 matches in our application.

Thus, we propose a parallel implementation of the elastic-matching algorithm based on the techniques summarized in Section 2.

3.1 The 2D-FFT

Let $\omega_m := \exp(-2\pi i/m) \in \mathbb{C}$ be a root of unity and let $F_m := \frac{1}{\sqrt{m}}(\omega_m^{jk})_{j,k=0,\ldots,m-1} \in \mathbb{C}^{m\times m}$ be a Fourier-matrix. The two dimensional discrete Fourier-transformation of a matrix $B \in \mathbb{C}^{m\times n}$ with m rows and n columns is given by

$$\hat{B} = F_m \cdot B \cdot F_n.$$

The intermediate $B' := F_m \cdot B$ can be computed in terms of one-dimensional FFTs applied to the columns of B and the computation of $\hat{B} = B' \cdot F_n$ can be done in terms of one-dimensional FFTs applied to the rows of B'. Based on this observation one can deduce a fast two dimensional FFT (2D-FFT).

3.2 Parallelizing the 2D-FFT

Our implementation of the 2D-FFT essentially follows [7, §23.3.1]. However, modifications have been applied to the communication structure. The data, here an $m \times n$ matrix, is distributed over P processes. For ease of presentation we assume m, n, and P to be powers of two.

Assuming that the data is initially distributed as column blocks, each process can perform the column FFTs on its data, since the data is complete for each process.

For the row FFT however, data from every other process is needed. Thus, in a naive implementation an all-to-all communication is unavoidable.

In our implementation these P^2 send operations are scheduled such that at P different time-steps only P messages have to be sent. To this end, we introduce column and row processes which compute the FFT of the columns and rows, respectively. We distribute the data such that the row processes can receive portions of their data while the column processes are still in a computing phase. At any time-step, every column process sends to exactly one row process, and every row process receives from exactly one column process. Thus, the communication and computation time overlap. Moreover, the data sent from a column process can be received directly, no data is left in the network.

3.3 Details of the algorithm

A standard FFT algorithm for $x = (x_\ell)_{\ell=0}^{m-1} \in \mathbb{C}^m$, for m a power of two, is summarized in Tab. 3. Note, for ease of presentation we used a naive way of computing the roots of unity. In our implementation, these roots are precomputed and supplied to the FFT algorithm, see also [21].

The first step in the generic case $q > 1$ is to split the input data x into $u = (u_\ell)_{\ell=0}^{q-1}$ and $v = (v_\ell)_{\ell=0}^{q-1}$, where $q = m/2$, $u_\ell = x_\ell + x_{\ell+q}$, and $v_\ell = \omega_q^\ell (x_\ell - x_{\ell+q})$. Next the FFT is applied to u and v leading to $\hat{u} = F_q u$ and $\hat{v} = F_q v$. With $\hat{u}$ all even coefficients of $\hat{x}$ are known and with $\hat{v}$ all odd coefficients of $\hat{x}$ are known.

The key observation is that the computation order of the odd/even part in the recurrence is arbitrary. Moreover, the data of the row processes can be organized according to the odd/even decomposition.

Suppose we have $P = 2$ column processes CP_0 and CP_1 and row processes RP_0 and RP_1. In time-step 1 CP_0 processes the even part of its data while CP_1 processes the odd part of its data. In time-step 2 CP_0 sends the processed even part to RP_0 and processes the odd part of its data while CP_1 sends the processed odd part of its data to RP_1 and processes the even part of its data. The communication is splitted over two time-steps and overloaded with computation time. Finally, CP_0 sends the processed odd part to RP_1 while CP_1 sends the processed even part to RP_0. Now, RP_0 and RP_1 have the complete data for the row FFT. The row FFT can be computed using the standard 1D-FFT, cf. Tab. 3.

This idea can easily be extended to more processes. In the j-th recursion of the FFT the parts $(x'_\ell)_{\ell \equiv k \bmod 2^j}$ of $x' = F_m \cdot x$ are computed, $k = 0, \ldots, 2^j - 1$. Although the computation of these parts is sequential it can be performed in any order. In particular, these parts can be computed in a different order on different processes.

In our implementation the column process numbers are used to assign a unique computation order $\sigma_{j,k}$ for any column process j, where $\sigma_{j,k}$ is a permutation of the numbers $0, \ldots, P-1$ for any j and for any k. The technique described in the following leads to an easy implementation of the parallel column FFT, see Tab. 4.

In the first time-step, the order odd/even in the j-th recurrence is determined by the j-th last bit of the binary representation of the column process number. Thus, the path followed initially can be viewed as a bit-reversal of the column process number. Following the binary structure of the 1D-FFT, the ordering for the k-th time-step can be obtained by an exclusive-or operation with the binary representation of the initial path and k, $k = 0, \ldots, P-1$. If, e.g., the initial path is 010, the ordering is $010 = 010 \oplus 000$, $011 = 010 \oplus 001$, $000 = 010 \oplus 010$ etc. The destination row process for the processed data is given by the path followed in the current time-step. Row process RP_k receives the data x'_ℓ with $\ell = k \bmod P$, $k = 0, \ldots, P-1$.

Example: Suppose $P = 4$. From the binary representation 00, 01, 10, and 11 it follows that CP_0, CP_1, CP_2, and CP_3 initially compute the 00, 10, 01, and 11 part, respectively, of the data (0/1 is related to odd/even). The binary representation of the computation order for CP_0 is 00, 01, 10, and 11. RP_0 receives successively from CP_0, CP_2, CP_1, and CP_3. A complete ordering diagram for this case is shown in Fig. 2.

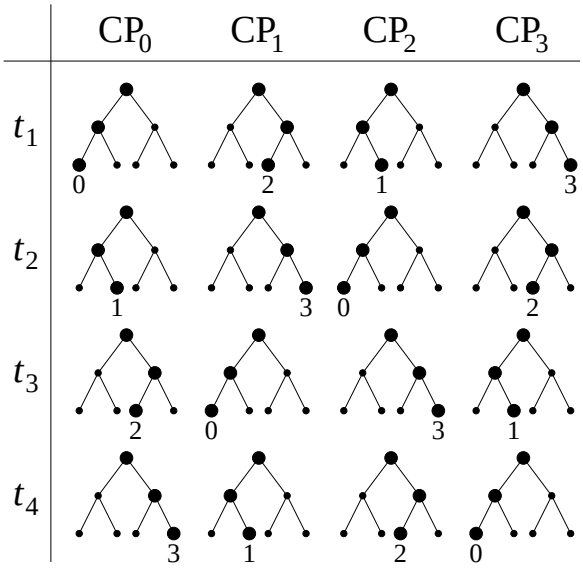

Figure 2: Communication diagram for $P = 4$; the rows show the four time-steps t_j, $j = 1, 2, 3, 4$, the columns show the current recurrence path of the four column processes (left/right: computation of the odd/even part in the recurrence); the pathes define the destination row processes.

In our implementation this data-flow is controlled by the numbers `rec_order` (essentially the j last bits of the column process number) and `dest_proc` (the number of the destination row process), see Tab. 4.

The data for the row process k contains the rows $\rho \equiv k \bmod P$ instead of m/P consecutive rows. Thus, finishing the $\log_2(P)$-th recurrence, the data for the row processes is completed and a standard 1D-FFT can be used for computing the row FFT.

The overall algorithm for the column FFT is summarized in Tab. 4, an extension to multiple column input is straightforward. An initial call of this function by CP_j is $\hat{x} =$ `ColumnFFT`$(x, j, 0)$. This completes the description of our 2D-FFT implementation.

The parallelization of the point-wise multiplication in Tab. 2 is straightforward. The inverse 2D-FFT can be derived from the 2D-FFT by replacing ω_q by its complex conjugate $\bar{\omega}_q$.

4 Performance Measurements

In order to investigate the behavior of our parallel implementation we did performance measurements on a PC-Cluster. We employed a special message passing interface to enable access to high speed networks and to preserve portability. We ran experiments with different image sizes (256×256, 512×512, and 1024×1024 pixel images).

4.1 The Störtebeker PC-Cluster

The Störtebeker Cluster at the University of Lübeck is a tightly coupled PC-Cluster with a high speed network interconnection. The cluster is built from 48 dual processor PCs with Pentium II (333 MHz) and 128 MBytes or 256 MBytes memory per node, respectively. All nodes are connected by Myrinet, cf. [3, 16], via four 8-port single chip switches. Figure 3 shows the topology for this interconnection. Myrinet is a proprietary product which uses parallel data transmission over 20 individually shielded wires. The nominal bandwidth of Myrinet is 1.25 Gbit/s and it has a hardware latency of 3 μs [17]. Recently, it has gained high popularity in cluster systems. Apart from the good price-performance ratio this is caused by its flexible network access via a dedicated communication processor.

For administrative purposes the nodes of the cluster are connected via Fast Ethernet. Linux 2.2.10 is used as the operating system on all nodes.

4.2 Message Passing Environment

In order to enable efficient direct access without losing portability of the applications, our lean message passing programming interface HPCC (High Performance Cluster Communication) was employed [17]. It acts as a common abstraction layer for various high speed network technologies.

Table 3: Standard 1D-FFT implementation.

```
function x̂ = FFT(x)
  q = length(x)/2; ω_q = exp(−2πi/q);
if q > 1,
  (u_ℓ)_{ℓ=0}^{q−1} := (x_ℓ + x_{ℓ+q})_{ℓ=0}^{q−1};
  (v_ℓ)_{ℓ=0}^{q−1} := ω_q^ℓ · (x_ℓ − x_{ℓ+q})_{ℓ=0}^{q−1};
  û := FFT(u);
  v̂ := FFT(v);
  (x̂_{2ℓ})_{ℓ=0}^{q−1} := (û_ℓ)_{ℓ=0}^{q−1};
  (x̂_{2ℓ+1})_{ℓ=0}^{q−1} := (v̂_ℓ)_{ℓ=0}^{q−1};
else
  x̂_0 := x_0 + x_1; x̂_1 := x_0 − x_1;
end.
```

Table 4: Parallel 1D-FFT implementation.

```
function x̂ = ColumnFFT(x, rec_order, dest_proc)
  q = length(x)/2; ω_q = exp(−2πi/q);
if q ≤ m/P,
  x̂ = FFT(x); send x̂ to dest_proc;
else
  (u_ℓ)_{ℓ=0}^{q−1} := (x_ℓ + x_{ℓ+q})_{ℓ=0}^{q−1};
  (v_ℓ)_{ℓ=0}^{q−1} := ω_q^ℓ · (x_ℓ − x_{ℓ+q})_{ℓ=0}^{q−1};
  if rec_order ≡ 0 mod 2,
    ColumnFFT(u, rec_order ÷ 2, 2 · dest_proc);
    ColumnFFT(v, rec_order ÷ 2, 2 · dest_proc + 1);
  else
    ColumnFFT(v, rec_order ÷ 2, 2 · dest_proc + 1);
    ColumnFFT(u, rec_order ÷ 2, 2 · dest_proc);
  end
end.
```

Currently, direct access to four high speed networks is supported: HIC (Heterogeneous Interconnect) [13]; SCI (Scalable Coherent Interface) [14, 20]; Myrinet [3]; and Gigabit Ethernet [17]. Furthermore, indirect access to IP (Internet Protocol) based networks is provided, e.g., for developing applications on clusters without a

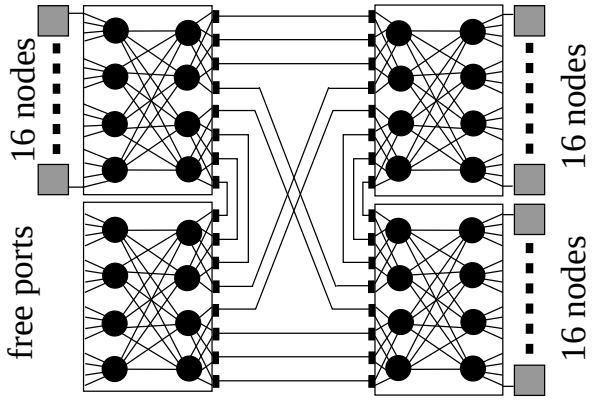

Figure 3: Topology for Myrinet interconnection of "Störtebeker Cluster".

high speed network. This variant uses PVM (Parallel Virtual Machine) [9] as the underlying communication platform.

HPCC provides the basic functionality for communication (connection establishment, send, receive) and dynamic process creation. In order to exploit the performance of high speed devices, unbuffered send and receive operations with zero-copy are offered besides the usual buffered, blocking send and receive operations. The uniform access to different device technologies via HPCC allows comparative performance measurements for all supported networks with the same, unmodified program. More details of the HPCC implementation were described in previous work [17].

4.3 Measurements

We tested the performance of our parallel implementation by running experiments using different numbers of processing nodes (2, 4, 8, 16, and 32) on our PC-Cluster. To avoid limiting the number of processes per node we used IP based interconnections with both Myrinet and Fast Ethernet (FE). We calculated the average iteration time over 50 iterations. Each experiment was performed several times.

Figure 4 shows the average iteration time for different image sizes on a logarithmic scale. Using the Myrinet network interconnection our parallel FFT-techniques scales very well. By doubling the number of processing nodes the average iteration time is nearly halved. This can be observed even for a large number of processing nodes. With Fast Ethernet, the parallel algorithm also works quite well. However, as the number of processing nodes increases beyond a certain point, the performance collapses for small images. This is due to the fact that the Fast Ethernet network provides unsatisfactory infrastructure for cluster computing. Fast Ethernet uses a shared medium, so a large amount of communication results in traffic congestions because of many collisions. This leads to high latency which handicaps the message passing environment. For an image size of 256×256 and 32 nodes, so much bandwidth is lost through collisions that

the running time actually increases compared to the experiment with 16 nodes. Our application is particularly sensitive to these effects since it requires a lot of bandwidth to exchange the data between the row and column processes.

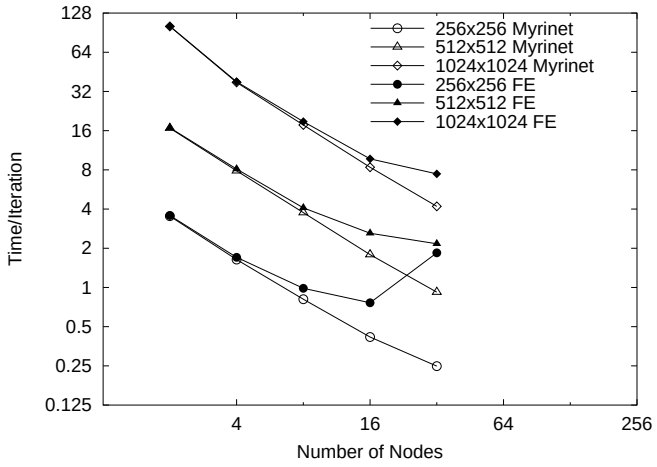

Figure 4: Average running time versus number of processing nodes (logarithmic scale).

5 Conclusions

We proposed a parallel implementation of the elastic-matching algorithm based on [8]. Using FFT type techniques, the numerical complexity for the solution of the linear system can be reduced from $\mathcal{O}(N^3)$ using standard techniques to $\mathcal{O}(N \log N)$ where N denotes the number of pixels.

Although the FFT is an outstanding method in a sequential architecture, its parallelization on a distributed memory machine provides some severe difficulties. We overcome these difficulties by introducing an appropriate communication strategy. With this strategy, we end up with an almost linear speed-up in the Myrinet environment and for a realistic number of nodes. The overall algorithm enables the registration of large images and registration sequences of many images in a reasonable time.

Thus, we see this algorithm as a promising starting point for further investigation, e.g. with respect to multiscale approaches, 3D registration, or multi-modal image registration.

Acknowledgments

The authors are indebted to Oliver Schmitt from the Institute of Anatomy of the Medical University of Lübeck, for providing all the anatomical data and for various interesting and inspiring discussions.

References

[1] Y. Amit, U. Grenander, and M. Piccioni, Structural Image Restoration Through Deformable Templates, Journal of the American Statistical Association, 86(414) (1991) 376–387.

[2] R. Bajcsy and S. Kovačič, Toward an Individualized Brain Atlas Elastic Matching, MS-CIS-86-71 Grasp Lap 76, Dept. of Computer and Information Science, Moore School, University of Philadelphia (1986).

[3] N. J. Boden, D. Cohen, R. E. Felderman, A. E. Kulawik, C. L. Seitz, J. N. Seizovic, and W.-K. Su. Myrinet: A gigabit-per-second Local Area Network. *IEEE Micro*, 15(1):29–36, Feb. 1995.

[4] Böhme, M. and B. Siebert, Parallele 2-dimensionale Fast Fourier-Transformation zum Elastic Matching, Teil 2 (Programmdokumentation). Technical Report B-00-02, Med. Univ. Lübeck, 2000.

[5] M. Bro-Nielsen, Medical Image Registration and Surgery Simulation, Ph.D. thesis, IMM, Technical University of Denmark (1996).

[6] G.E. Christensen, Deformable Shape Models for Anatomy, Ph.D. thesis, Sever Institute of Technology, Washington University (1994).

[7] Chu, E. and A. George, *Inside the FFT Black Box*. CRC Press, 2000.

[8] Fischer, B. and J. Modersitzki, Fast inversion of matrices arising in image processing. *Numerical Algorithms*, 22:1–11, 1999.

[9] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V. Sunderam. *PVM: Parallel Virtual Machine – A Users' Guide and Tutorial for Networked Parallel Computing*. The MIT Press, 1994.

[10] Golub G.H. and C.F. van Loan, *Matrix Computations*. John Hopkins University Press, 1996.

[11] C. Grewe, W. Obelöer, S. Petri, and M. Boosten. Network Interface Software Development for Medical Applications. ESPRIT Project 20693 "ARCHES" Deliverable 3.4.1, Institut für Technische Informatik der MU Lübeck, Dec. 1998.

[12] M.E. Gurtin, *An Introduction to Continuum Mechanics*, Academic Press, Orlando (1981).

[13] IEEE Standard for Heterogeneous InterConnect (HIC). IEEE Standard 1355-1995, IEEE Standards Board, 1995.

[14] IEEE Standard for Scalable Coherent Interface (SCI). IEEE Standard 1596-1992, IEEE Standards Board, 1993.

[15] Modersitzki J. and O. Schmitt, Image registration of histological sections. Technical Report A-00-06, Med. Univ. Lübeck, 2000.

[16] Myricom. The GM Message Passing System. WWW page, Aug. 1998. `http://www.myri.com/GM/doc/gm_toc.html`.

[17] S. Petri, G. Lustig, C. Grewe, R. Hagenau, W. Obelöer, and M. Boosten. Performance Comparison of different High-Speed Networks with a Uniform Efficient Programming Interface. In W. Karl and G. Horn, editors, *Scalable Coherent Interface – Proceedings of SCI Europe '99*, pp. 83–90, Sept. 1999.

[18] Schormann, T., von Matthey, M., Dabringhaus, A., Zilles, K.: Alignment of 3-D Brain Data Sets Originating from MR and Histology, Bioimaging, Vol. 1 (1993) 119–128

[19] T. Schormann and K. Zilles, "Limitation of the principal axes theory," *IEEE Transaction on medical imaging*, vol. 16, pp. 924–947, 1997.

[20] H. Hellwagner and A. Reinefeld, editors. *Scalable Coherent Interface: Technology and Applications – Proceedings of SCI Europe '98*. Cheshire Henbury, Sept. 1998.

[21] Tasche, M. and Zeuner, H.: Worst and average case roundoff error analysis for FFT, BIT Numerical Mathematics, Vol. 41, Issue 3 (September 2001).

[22] J.-P. Thirion, Non-Rigid Matching Using Demons, *in IEEE, Correspondence on Computer Vision and Pattern Recognition*, 1996, 245–251.