Parallelisierung eines nichtlinearen Registrierungsalgorithmus zur Verarbeitung sehr großer Volumen-Daten

Matthias Bolten[1], Nils Papenberg[1], Bernd Fischer[1], Panagiotis Adamidis[2], Rolf Rabenseifner[2] und Holger Berger[3]

[1]SAFIR, Institut für Mathematik, Universität zu Lübeck, 23560 Lübeck
[2]Höchstleistungsrechenzentrum Stuttgart, 70550 Stuttgart
[3]NEC High Performance Computing Europe GmbH, 70565 Stuttgart
Email: [bolten,papenber,fischer]@math.uni-luebeck.de,
[rabenseifner,adamidis]@hlrs.de, hberger@hpce.nec.com

Zusammenfassung. Nichtlineare Registrierung ist eine der rechenintensivsten Aufgaben in der Bildregistrierung. Dies verbietet oftmals einen Einsatz dieser Methoden in zeitkritischen Anwendungen im klinischen Einsatz. In dieser Arbeit wird eine sehr effiziente parallelisierte und vektorisierte Implementierung des diffusiven Registrierungsalgorithmus vorgestellt. Testläufe auf Höchstleistungsrechnern haben gezeigt, dass damit die Durchführung einer Iteration des Verfahrens auch bei Datensätzen mit 511^3 Voxeln in weniger als einer Sekunde möglich ist. Damit stehen diese Methoden auch bei hochauflösenden Daten für zeitkritische Anwendungen zur Verfügung.

1 Einleitung

Die Registrierung zweier Bilder ist ein häufig auftretendes Problem in der medizinischen Bildverarbeitung. Bei der Registrierung ist eine Beschreibung der Deformation gesucht, die ein Bild in ein gegebenes anderes transformiert. Diese Deformation benötigt man in der Medizin unter anderem um Atlas-Informationen auf ein zweites Bild zu übertragen oder um Therapieverläufe zu analysieren.

Neben rigiden Methoden haben sich rechenintensivere nicht-rigide Verfahren etabliert. Die nicht-rigiden Verfahren liefern deutlich bessere Registrierungsergebnisse als die rigiden Verfahren. Populäre Vertreter der nicht-rigiden Verfahren sind beispielsweise die elastische Registrierung, die fluidale Registrierung oder die hier betrachtete diffusive Registrierung (siehe [1]). Die algorithmische Komplexität dieser Verfahren verbietet ihren Einsatz aber oftmals, da z.B. bei intraoperativem Einsatz das Ergebniss in möglichst kurzer Zeit geliefert werden muss.

Da nicht-rigide Bildregistrierung ein aufwändiger Vorgang ist, sind in der Vergangenheit verschiedene Ansätze parallelisiert worden, etwa der Dämonenansatz [2] oder die elastische Registrierung [3]. In dieser Arbeit wurde der diffusive Registrierungsalgorithmus parallelisiert und vektorisiert, für den bereits ein schneller

serieller Algorithmus, der $\mathcal{O}(N)$ arithmetische Operationen pro Iterationsschritt benötigt, zur Verfügung steht. Dadurch steht nun ein Werkzeug zur Verfügung, das ein nicht-rigides Registrierungsverfahren auch für zeitkritische Applikationen zur Verfügung stellt.

2 Diffusive Registrierung

Als Grundlage dieser Arbeit wurde die diffusive Registrierung gewählt. Zum besseren Verständniss wird hier zunächst die variationelle Formulierung und die numerische Lösung des Registrierungsproblems mit der diffusiven Registrierung skizziert. Weitergehende Informationen findet man in [1, 4, 5].

2.1 Variationelle Formulierung

Mathematisch lässt sich das Registrierungsproblem folgendermaßen formulieren. Seien $R, T : \Omega \to \mathbb{R}$, $\Omega \subset \mathbb{R}^d$, d-dimensionale Bilder. Gesucht ist ein Verrückungsfeld $\mathbf{u} : \mathbb{R}^d \to \mathbb{R}^d$, so dass der Abstand $\mathcal{D}$ zwischen $T(\mathbf{x} - \mathbf{u}(\mathbf{x}))$ und $R(\mathbf{x})$ minimal wird. Da in der medizinischen Anwendung die Glattheit von $\mathbf{u}$ gewünscht ist, wird ein Regularisierer $\mathcal{S}$, dessen Einfluss über α gesteuert wird, eingeführt und man erhält das folgende Funktional zur Minimierung:

$$\mathcal{J}[\mathbf{u}] = \mathcal{D}[R, T; \mathbf{u}] + \alpha \mathcal{S}[\mathbf{u}]. \tag{1}$$

Mit Hilfe der Variationsrechnung kann in Abhängigkeit von $\mathcal{D}$ und $\mathcal{S}$ ein System von partiellen Differentialgleichungen, die *Euler-Lagrange-Gleichungen*, aufgestellt werden, die numerisch gelöst werden können, um $\mathbf{u}$ zu bestimmen.

Es wurde die bekannte *sum of squared differences* (*SSD*) als Distanzmaß und der Regularisierer der diffusiven Registrierung gewählt:

$$\mathcal{D}^{SSD}[R, T; \mathbf{u}] = \int_{\Omega} \left(R(\mathbf{x}) - T(\mathbf{x} - \mathbf{u}(\mathbf{x}))\right)^2 d\mathbf{x},$$

$$\mathcal{S}^{diff}[\mathbf{u}] = \frac{1}{2} \sum_{l=1}^{d} \int_{\Omega} ||\nabla \mathbf{u}_l||_2^2 d\mathbf{x}.$$

Damit $\mathbf{u}$ das Funktional (1) minimiert, muss $\mathbf{u}$ die Euler-Lagrange-Gleichungen erfüllen:

$$\mathbf{f}^{SSD}(\mathbf{x}, \mathbf{u}(\mathbf{x})) + \alpha \mathcal{A}^{diff}[\mathbf{u}](\mathbf{x}) = 0, \ \mathbf{x} \in \Omega$$

mit $\mathbf{f}^{SSD}(\mathbf{x}, \mathbf{u}(\mathbf{x})) = (R(\mathbf{x}) - T(\mathbf{x} - \mathbf{u}(\mathbf{x}))) \cdot \nabla T(\mathbf{x} - \mathbf{u}(\mathbf{x}))$ und $\mathcal{A}^{diff}[\mathbf{u}] = \Delta \mathbf{u}$.

2.2 Numerische Lösung

Das erhaltene System wurde mit Hilfe von finiten Differenzen diskretisiert und mit einem Zeitschrittverfahren der Form

$$\partial_t \mathbf{u}^{(k+1)}(\mathbf{x}, t) - \alpha \mathcal{A}^{diff}[\mathbf{u}^{(k+1)}](\mathbf{x}, t) = \mathbf{f}^{SSD}(\mathbf{x}, \mathbf{u}^{(k)}(\mathbf{x}, t)), \ k \geq 0$$

gelöst. Durch die Diskretisierung erhält man eine Matrix-Repräsentation $\mathbf{A}$ von $\partial_t - \alpha \mathcal{A}^{diff}$. Zur Bestimmung der neuen Iterierten $\mathbf{u}^{(k+1)}$ muss ein lineares Gleichungssystem der Form

$$\mathbf{A}\mathbf{u}^{(k+1)} = \mathbf{f}^{(k)} \tag{2}$$

gelöst werden. Im Falle der diffusiven Registrierung lässt sich dieses Gleichungssystem mit Hilfe eines AOS-Schemas [4] und des Thomas-Algorithmus [6] sehr schnell lösen. Im Unterschied zur herkömmlichen Lösung wird das große Gleichungssystem (2) in kleine Gleichungsysteme aufgeteilt. Dies bedeutet für den 3D-Fall mit Bildern der Größe $m \times n \times o$, dass nicht mehr ein System mit $3*m*n*o$ Gleichungen, sondern $3*n*o$ Systeme mit m Gleichungen in x-Richtung, $3*m*o$ Systeme mit n Gleichungen in y-Richtung und $3*m*n$ Systeme mit o Gleichungen in z-Richtung zu lösen sind. Diese kleineren Gleichungssysteme sind nur noch tridiagonal. Mit dem Thomas-Algorithmus [6] steht ein schneller $\mathcal{O}(N)$-Löser für tridiagonale Gleichungssysteme zur Verfügung. Näheres zum AOS-Schema und dem Einsatz in der diffusiven Registrierung findet sich in [4].

3 Parallelisierung

Bei der numerischen Lösung muss in jedem Schritt die Kraft $\mathbf{f}$ als rechte Seite und die Lösung des linearen Gleichungssystems (2) bestimmt werden. Da die Kraftberechnung punktweise mit Benutzung von Nachbarschaftsinformationen erfolgt, ist die Parallelisierung dieses Schrittes trivial. Die Parallelisierung des Lösens des Gleichungssystems stellt eine größere Herausforderung dar. Der verwendete Thomas-Algorithmus ist inhärent sequentiell, da für jede Komponente die jeweils vorhergehende Komponente benötigt wird. Der Thomas-Algorithmus selbst ist also nicht parallelisierbar. Dieses Problem ist jedoch zu umgehen, da viele tridiagonale Gleichungssysteme gelöst werden müssen. Folglich werden die jeweils zu lösenden Gleichungssysteme in x-, y- bzw. z-Richtung auf die einzelnen Prozessoren verteilt. Die Zielplattform hat gemeinsamen Speicher, so dass hier keine Probleme auftraten.

Auf Maschinen mit verteiltem Speicher benötigt man einen anderen Ansatz. Eine Möglichkeit ist die Verteilung der Daten auf die einzelnen Prozessoren, so dass die Gleichungssysteme in x-Richtung und y-Richtung unabhängig voneinander gelöst werden können. Für die Lösung in z-Richtung werden die Daten dann umverteilt. Dieser Ansatz wird für ein AOS-Schema in [7] verfolgt.

4 Implementierung

Die Implementierung des Algorithmus erfolgte in C. Das C-Programm wurde dann wie oben beschrieben mit OpenMP parallelisiert und mit Hilfe der Compilerdirektiven des NEC-Compilers vektorisiert. Bei der Vektorisierung wurde die beschriebene Idee zur Parallelisierung auf den einzelnen Prozessoren ausgenutzt, so dass zwei verschiedene Ebenen von Parallelität im Programm existieren: Eine grobe Verteilung auf die einzelnen Prozessoren und eine feine Verteilung, um die

Tabelle 1. Benötigte Zeit für 50 Iterationen der Fixpunktiteration auf einer NEC SX-5 bei Eingabe eines Bildes mit 511^3 Voxeln

#CPUs	Gesamtzeit	**Zeit/Iteration**	Mflops/CPU	Mflops gesamt	Speedup	Effizienz
1	486 s	**9,72 s**	1307	1307	1,0	-
8	75 s	**1,50 s**	1059	8469	6,5	0,81
16	49s	**0,98 s**	810	12963	9,9	0,62

Vektorprozessoren auszunutzen. Als Zielplattform wurde die NEC SX-6/48M6 beim Höchsteleistungsrechenzentrum Stuttgart ausgewählt, da es sich um eine Vektormaschine handelt. Der Einsatz eines Vektorrechners bietet sich an, da die Art des zugrundeliegenden Algorithmus, der in sehr kurzer Zeit das gesammte Datenvolumen in einer bestimmten Reihenfolge bearbeiten muss, sehr Cache-unfreundlich ist. Bei ccNUMA-Maschinen wäre neben diesem Problem zusätzlich zu beachten, dass das Programmiermodell zwar von gemeinsamen Speicher ausgeht, Speicher auf anderen physikalischen Knoten aber deutlich langsamer ist. Auch dieses Problem ergibt sich bei Vektorrechnern naturgemäß nicht.

5 Ergebnisse

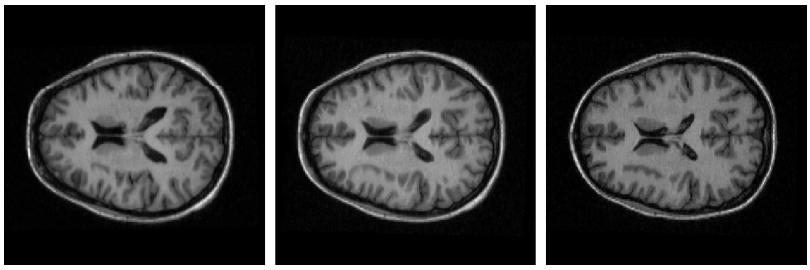
Die Implementierung wurde zunächst auf der NEC SX-5 des Höchstleistungsrechenzentrums Stuttgart (HLRS) getestet. Für die Zeitmessungen wurden zwei synthetische Datensätze mit jeweils 511^3 Voxeln registriert, wobei 50 Iterationen der Fixpunktiteration durchgeführt wurden. Bei dieser Größe muss in jeder Iteration ein lineares Gleichungssystem mit 4.000.298.493 Unbekannten gelöst werden, bzw. bei der Verwendung von AOS 783.363 Gleichungssysteme mit jeweils 511 Unbekannten. Die erhaltenen Zeiten sind in Tabelle 1 zu finden.

Zum Vergleich wurden die selben Bilder noch einmal auf dem Nachfolgemodell der SX-5, der NEC SX-6/48M6, am HLRS gerechnet. Bei der Verwendung von 8 CPUs auf dieser Maschine benötigte der Algorithmus nur noch 35 Sekunden für die 50 Iterationen, d.h. 0,70 Sekunden pro Iteration.

Die Bildgröße von 511^3 Voxeln wurde gewählt, um Performance-Einbußen zu vermeiden, da die NEC-Maschine 4096 Speicherbänke besitzt und die Anzahl der Speicherbänke nicht durch die Inkremente teilbar sein sollte, um mit der vollen Geschwindigkeit auf den Speicher zugreifen zu können. Ein Verzicht auf einen Voxel in jeder Dimension sollte aber kein Problem darstellen.

Zusätzlich zu den großen synthetischen Testdaten wurden kleinere Datensätze mit 256^3 Voxeln registriert, die aus den oben genannten Gründen auf die Größe von 255^3 reduziert wurden. Die Daten wurden freundlicherweise von Benoít Dawant (Vanderbilt University) zur Verfügung gestellt. Dieser Test soll dazu dienen, um zu zeigen, dass der Algorithmus praktisch anwendbar ist. Die Ergebnisse sind in Abbildung 1 zu finden.

Abb. 1. Schicht 128 des Referenzvolumens, des verrückten Templatevolumens nach 50 Iterationen und des unveränderten Templatevolumens. Die Ähnlichkeit ist insbesondere an der äußeren Schädelform und der Form der Ventrikel zu erkennen.



6 Fazit und Ausblick

Mit der hier vorgestellten Implementierung einer parallelen und vektorisierten Variante der diffusiven Registrierung steht ein Algorithmus zur Verfügung, mit dem sich auch sehr große klinisch relevante Bilddatensätze in sehr kurzer Zeit mit nicht-linearen Methoden registrieren lassen. Die erste Implementierung auf einem Vektorrechner ergab eine hervorragende Performance. Damit sind diese Werkzeuge auch in zeitkritischen Bereichen wie der intraoperativen Bildverarbeitung einsetzbar. In Zukunft werden wir weitere Regularisierer und Distanzmaße parallelisieren, um in vielen Anwendungsbereichen adäquate Werkzeuge zur schnellen Registrierung auf Parallelrechnern zur Verfügung zu haben. Zudem arbeiten wir neben der hier verwendeten Shared-Memory-Parallelisierung an parallelen Algorithmen für verteilte Systeme ohne gemeinsamen Speicher, um die Anzahl der verwendbaren Systeme zu erhöhen.

Literaturverzeichnis

1. Modersitzki J. Numerical Methods for Image Registration. Numerical Mathematics and Scientific Computation. Oxford University Press; 2004.
2. Stefanescu R, Pennec X, Ayache N. Grid-enabled non-rigid registration of medical images. Parallel Processing Letters 2004;14(2):197–216.
3. Christensen GE. MIMD vs. SIMD Parallel Processing: A Case Study in 3D Medical Image Registration. Parallel Computing 1998;(24):1369–1383.
4. Fischer B, Modersitzki J. Fast diffusion registration. In: Nashed MZ, Scherzer O, editors. Inverse Problems, Image Analysis and Medical Imaging. vol. 313 of Contemporary Mathematics. AMS; 2002. p. 117–129.
5. Fischer B, Modersitzki J. A unified approach to fast image registration and a new curvature based registration technique. Linear Algebra and its Applications 2004;(380):107–124.
6. Ames WF. Numerical Methods for Partial Differential Equations, Second Edition. Academic Press, New York; 1977.
7. Bruhn A, Jakob T, Fischer M, et al. High performance cluster computing with 3-D nonlinear diffusion filters. Real-Time Imaging 2004;(10):41–51.