CUDA Optimierung von nicht-linearer oberflächen- und intensitätsbasierter Registrierung

Stefan Köhnen, Jan Ehrhardt, Alexander Schmidt-Richberg, Heinz Handels

Institut für Medizinische Informatik, Universität zu Lübeck
stefan.khnen@googlemail.com

Kurzfassung. Die vorliegende Arbeit beschäftigt sich mit der Implementierung von Teilen eines Registrierungsalgorithmus in der Compute Unified Device Architecture (CUDA) von NVIDIA und der daraus resultierenden Zeitersparnis. Es wurden die einzelnen Schritte des Registrierungsalgorithmus analysiert und auf ihre Parallelisierbarkeit untersucht. Die Implementierungen wurden anhand von 20 thorakalen CT-Datensätzen evaluiert und der SpeedUp berechnet. Es wurde eine Beschleunigung vom Faktor 143 bei der TPS Interpolation und ein Faktor 12 beim Image Warping erreicht. Obwohl nur 2 Teilschritte auf der GPU umgesetzt wurden, konnte ein Speedup des Gesamtverfahren von 2.175 erreicht werden. Dies zeigt das eine GPU-Implementierung effizienter als eine CPU-basierte Parallelisierung sein kann.

1 Einleitung

Registrierungsalgorithmen haben zahlreiche Anwendungen im Bereich der Medizinischen Bildverarbeitung, sind jedoch insbesondere bei der Verarbeitung großer Volumendatensätze mit einem erheblichen Rechenaufwand verbunden. In vorherigen Arbeiten [1, 2] wurde ein nicht-linearer Registrierungsalgorithmus entwickelt und z.B. für die Schätzung von Organbewegungen oder die Registrierung thorakaler und abdominaler Organe verwendet. Verschiedene Studien und Anwendungen belegen die Genauigkeit und Robustheit dieses Verfahrens [2]. Trotz einer effizienten Implementierung und CPU-basierter Parallelisierung des Algorithmus, bleibt ein wesentlicher Nachteil des Verfahrens die benötigte Rechenzeit.

In verschiedenen Arbeiten wurden die Möglichkeiten zur Verwendung von Grafikprozessoren für generelle Berechnungen, z.B. mittels der NVIDIA CUDA API, genutzt, um erhebliche SpeedUps zu erzielen. So wurden z.B. auch verschiedene lineare und nicht-lineare Registrierungsansätze für Standard-Grafikhardware parallelisiert und dadurch erheblich beschleunigt [3, 4]. Im Rahmen des hier vorgestellten Projektes soll der in [2] vorgestellte Algorithmus unter Verwendung von Standard-Grafikhardware beschleunigt werden. Im Gegensatz zu [4] soll dabei die NVIDIA CUDA API verwendet werden. Da sich der Registrierungsansatz wesentlich von dem in [3, 4] parallelisierten Verfahren unterscheidet, kann auch auf die dort verwendeten Methoden nicht oder nur eingeschränkt zurückgegriffen werden.

2 Methoden

Ausgangsdaten für die Registrierung sind ein Referenzbild I_F und ein zu registrierendes Bild I_M, sowie zugehörige Segmentierungsmasken des relevanten Organs (z.B. der Lunge) M_F und M_M. Der Registrierungsalgorithmus besteht aus zwei wesentlichen Komponenten: einer oberflächenbasierten Vorregistrierung und einem diffeomorphen, intensitätsbasierten Registrierungsschritt [2]. Die einzelnen Teilschritte des Verfahrens sind in Abb. 1 zusammengefasst. Aus den Masken M_F und M_M werden Oberflächenmodelle generiert, die zunächst affin (mittels ICP) und anschließend nicht-linear registriert werden. Auf den registrierten Oberflächen werden korrespondierende Punkte gesampelt, um daraus mittels einer Thin-Plate-Spline Interpolation das Deformationsfeld φ^{pre} zu erzeugen. Die diffeomorphe, intensitätsbasierte Registrierung wird anschließend auf das Referenzbild I_F und $I_M \circ \varphi^{\text{pre}}$ angewendet. Die gesuchte Gesamttransformation ergibt sich dann durch die Konkatenation $\varphi = \varphi^{\text{diff}} \circ \varphi^{\text{pre}}$.

Für die Umsetzung der GPU-basierten Parallelisierung wurden zunächst die anteiligen Rechenzeiten der einzelnen Teilschritte bestimmt (Abb. 1). Anschließend wurde die Eignung der einzelnen Teilschritte für eine GPU-basierte Parallelisierung untersucht (Abschn. 2.1) und eine Gewichtung hinsichtlich Eignung und Relevanz für eine Beschleunigung festgelegt. Die GPU-basierte Parallelisierung wurde dann schrittweise entsprechend der gefundenen Gewichtung durchgeführt (Abschn. 2.2). Die GPU-Implementierung wurde für eine NVIDIA Quadro FX 3800 optimiert. Diese Grafikkarte hat 1024 MB globalen Speicher, 24 Multiprozessoren mit je 8 Prozessor-Kernen.

2.1 Laufzeitkomplexität und Parallelisierbarkeit

Die Schritte des Registrierungsalgorithmus wurden in der Reihenfolge der größtmöglichen Laufzeitersparnis analysiert. Die TPS Interpolation und die diffeomorphe Registrierung sind mit 47.8% und 45.0% die zwei aufwändigsten Schritte des Registrierungsalgorithmus und bieten sich damit für eine Parallelisierung an. Die TPS Interpolation kann aufgrund der einfachen Verarbeitungsschritte einfach in CUDA implementiert werden. Die diffeomorphe Registrierung besteht aus mehreren Teilen (Abb. 1): Die Exponentiation erfordert es, dass zwei Deformationsfelder im Grafikkartenspeicher gehalten werden. Das ist momentan aufgrund von Speichereinschränkungen nicht möglich, es wäre notwendig einen Streaming-Ansatz zu implementieren, um diese Einschränkung zu umgehen. Die Regularisierung wurde bereits stark auf der CPU optimiert und es ist fraglich in wieweit mit CUDA eine weitere Verbesserung erreicht werden kann. Das Warping ist aufgrund seiner simplen Funktion einfach in CUDA zu implementieren. Ein weiterer positiver Aspekt der CUDA Implementierung des Warping ist die Verwendung des Textur-Speicher, der es erlaubt die in der Grafikkarten-Hardware implementierte trilineare Interpolation zu nutzen. Die Berechnung des Update-Feld ist ein Schritt der sich aufgrund der einfachen Funktion gut mit CUDA implementieren läßt.

Die Umsetzung der Nicht linearen Oberflächen Registrierung und des ICP-Algorithmus benötigen eine Implementierung des k-Nearest-Neighbour-Algorithmus (kNN) auf der GPU. Zwar gibt es bereits Ansätze diesen Algorithmus in CUDA zu implementieren, diese sind aber in der Anzahl der verwendbaren Punkte begrenzt [5] und müssen daher zunächst angepasst werden. Die Oberflächengenerierung basiert auf dem Marching-Cubes-Algorithmus für den es bereits funktionierende Implementierungen in CUDA [6] gibt. Die Konkatenation hat nur einen geringen Anteil an der Gesamtlaufzeit, außerdem ist es wie bei der Exponentiation notwendig zwei Deformationsfelder im Grafikkartenspeicher zu halten.

Aufgrund obiger Analyse werden zunächst die Schritte TPS, Warping, Compute Update Field und Regularisierung in CUDA umgesetzt. Die Oberflächen Generierung ist aufgrund des geringen Anteils an der Laufzeit und dem hohem Aufwand einen Marching-Cubes-Algorithmus in CUDA zu implementieren nicht für eine CUDA-Implementierung vorgesehen. Da ICP und die nicht-lineare Oberflächen Registrierung den kNN-Algorithmus benötigen, muss zunächst eine kNN-Implementierung verfügbar sein die nicht in der Anzahl der Punkte limitiert ist. Die Schritte Exponantiation und Konkatenation können, aufgrund ihres hohen Speicherbedarfs, erst implementiert werden wenn ein Streaming-Ansatz für CUDA implementiert wurde.

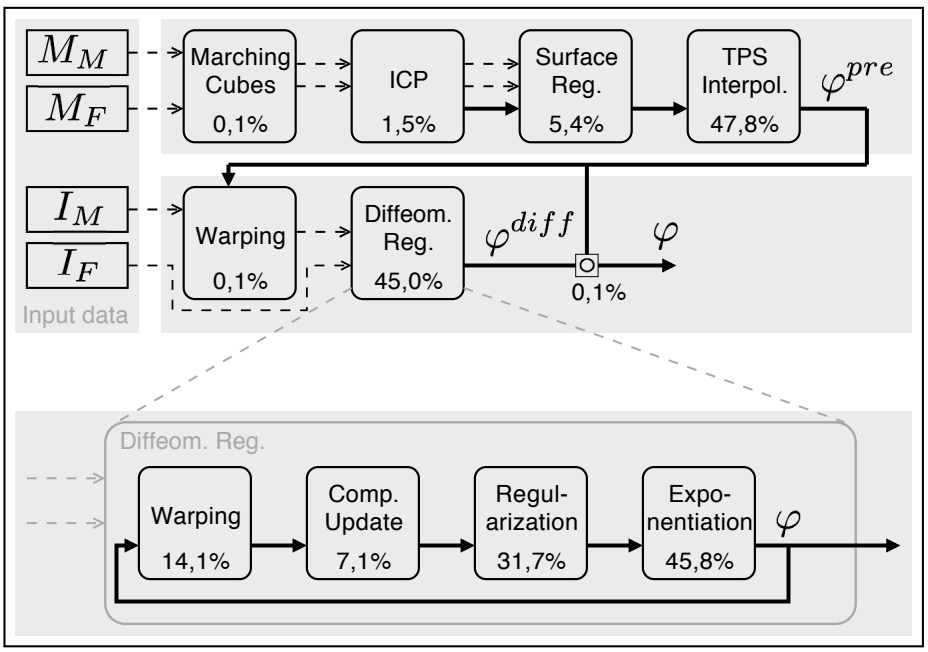


Abb. 1. Aufbau des Registrierungsalgorithmus. An jedem Schritt sind die Anteile an der Gesamtlaufzeit angegeben.

2.2 GPU-basierte Beschleunigung/Parallelisierung

Bei der TPS Interpolation wird an zwei korrespondierenden Landmarken Sets mit N Punkten ein Deformationsfeld bestimmt. Im ersten Schritt wird dabei eine Matrix der Größe $3 \times (3+N+1)$ auf der CPU invertiert. Die Matrix wird zusammen mit den Landmarken auf die GPU übertragen. Außerdem muss ein Bereich des Grafikkarten-Speichers für das zu generierende Deformationsfeld reserviert werden. Anschließend wird für jedes Voxel im Displacement Field ein Thread auf der Grafikkarte erzeugt, die dann parallel die Transformation berechnen.

Das Image Warping erfordert es das verwendete Bild und das Deformationsfeld in den Grafikspeicher zu übertragen. Um weitere Zeitersparnis zu erreichen, wird das Bild in den Textur-Speicher der Grafikkarte geladen und die Hardware implementierte Interpolation verwendet.

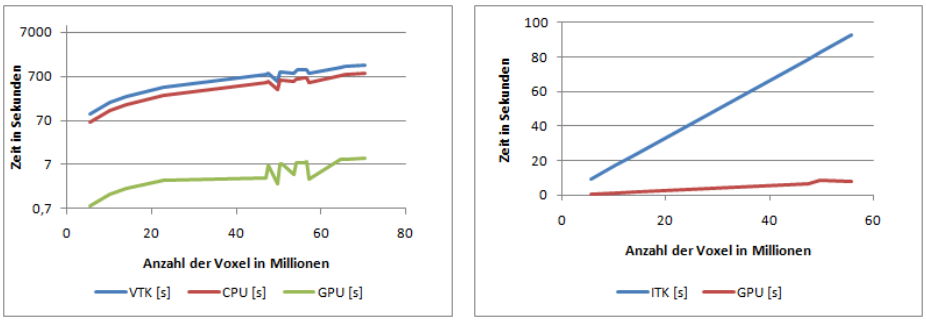
3 Ergebnisse

Die Laufzeiten der GPU-Programme wurden mit CudaEvents der NVIDIA CUDA API gemessen und umfassen auch die Operationen zur Speicherallokation und Speichertransfer. Die Laufzeiten der Programmteile die die CPU verwenden wurden mit der C time Bibliothek erfasst. Zur Messung wurden 20 thorakale CT-Datensätzen verwendet. Alle Zeiten wurden fünfmal gemessen und jeweils der Mittelwert verwendet.

Da die ursprüngliche Implementierung der TPS-Interpolation auf VTK basiert, wurde zusätzlich noch eine CPU basierte Ausführung des Kernel Code angegeben um Overhead von VTK bei der Zeitmessung auszuschließen (Abb. 2a).

Da beim Schritt Image Warping aufgrund der Normalisierungsschritte für den Textur-Speicher zusätzlicher Aufwand entsteht, umfasst die Zeitmessung auch diese Schritte (Abb. 2b). Die Normalisierungsschritte sind erforderlich, da der Textur-Speicher nur im Bereich [0;1] funktioniert.

Zur Berechnung der erreichten Beschleunigung des gesamten Registrierungsalgorithmus nach Amdahl [7] wurden die prozentualen Anteile aus Abbildung 1



(a) TPS (b) Warping

Abb. 2. Vergleich der Implementierungen

verwendet. Die Beschleunigung der TPS-Interpolation um den gemittelten Faktor 143 und des Image Warping um den gemittelten Faktor 12 ergibt eine Beschleunigung des gesamten Algorithmus um 2,175.

4 Diskussion

Am Graphen in Abb. 2a kann man erkennen das die Implementierung des TPS-Resampling in CUDA eine erhebliche Zeitersparnis mit sich bringt. Der Beschleunigungsfaktor liegt bei ca. 143 im Vergleich zu einer Implementierung mit VTK. Die Implementierung des Warping hat eine geringere Ersparnis gebracht (Abb. 2b), der Beschleunigungsfaktor liegt bei ca. 12. Allerdings wird durch die häufige Verwendung des Warping, im Gesamtkontext des Registrierungsalgorithmus, die Laufzeit erneut bedeutend verringert.

Die weiteren Schritte beinhalten die Umsetzung der in Abschnitt 2.1 genannten Schritte, wie beispielsweise die Berechnung des Update Feldes. Außerdem werden die Schritte die nicht einfach parallelisierbar sind weiterhin auf mögliche Optimierungen untersucht. Eine weitere Möglichkeit für Optimierungen wären die Normalisierungsschritte des Warping. Die Normalisierung könnte ebenfalls auf der GPU ausgeführt werden.

Durch eine Grafikkarte die mehr globalen Speicher bietet wäre es möglich weitere Teilschritte auch ohne einen Streaming-Ansatz zu implementieren. Die Arbeit zeigt aber, dass sich auch durch die GPU-Implementierung weniger Teilschritte nicht-lineare Registrierungs-Algorithmen erheblich beschleunigen lassen.

Literaturverzeichnis

1. Ehrhardt J, Werner R, Schmidt-Richberg A, et al. Statistical modeling of 4D respiratory lung motion using diffeomorphic image registration. IEEE Trans Med Imaging. 2010; p. 1–1. (accepted).
2. Schmidt-Richberg A, Ehrhardt J, Werner R, et al. Diffeomorphic diffusion registration of lung CT images. Med Image Anal for the Clin Grand Challenge. 2010; p. 55–62.
3. Modat M, Ridgway GR, Taylor ZA, et al. Fast free-form deformation using graphics processing units. Comput Methods Programs Biomed. 2010;98(3):278–84.
4. Köhn J, Drexl F, Köonig M, et al. GPU accelerated image registration in two and three dimensions. In: Proc BVM. Springer; 2006. p. 261–5.
5. Garcia V, Debreuve E, Barlaud M. Fast k nearest neighbor search using GPU. In: Proc CVPR Workshop on Computer Vision on GPU. Anchorage, Alaska, USA; 2008. p. 1–14.
6. Nguyen H. GPU Gems 3. Addison-Wesley Professional; 2007.
7. Amdahl GM. Validity of the single processor approach to achieving large scale computing capabilities. Proc AFIPS (Spring). 1967; p. 483–5.