Mehrstufige zeit- und bewegungsabhängige Rauschreduktion in Echtzeit mittels CUDA

Elmar Bührle[1], Benjamin Keck[1,2], Stefan Böhm[3], Joachim Hornegger[1]

[1] Lehrstuhl für Mustererkennung (LME), Universität Erlangen-Nürnberg, Martensstr. 3, 91058 Erlangen elmar@buehrle.org, {keck, hornegger}@informatik.uni-erlangen.de

[2] Siemens Healthcare, CV, Medical Electronics & Imaging Solutions, Postfach 3260, 91050 Erlangen

[3] Siemens Healthcare, Imaging & IT, Angiography, Siemensstr. 1, 91301 Forchheim, stefan.boehm@siemens.com

Motivation

- 2D Vorverarbeitungsalgorithmen zur Bildverbesserung erfordern eine hohe Rechenleistung.
- Bildverbesserung ist ohne teure Spezialhardware bisher nicht in Echtzeit möglich.
- Moderne GPUs ermöglichen durch hochparallele Verarbeitung eine enorme Rechenleistung, was bereits 2004 für die Echtzeitbildverarbeitung ausgenutzt wurde [1].
- NVIDIAs CUDA bietet eine relativ einfache Programmierbarkeit ihrer Manycore-GPU-Architektur durch einige Erweiterungen der Programmiersprache C [2].
- In der Computertomographie wurde CUDA bereits erfolgreich für die Beschleunigung der Rekonstruktion eingesetzt [3].

Methoden

Die mehrstufige zeit- und bewegungsabhängige Rauschreduktion besteht aus mehreren Komponenten(vgl. Abb. 1):

- Dekomposition in eine Laplace-Auflösungshierarchie mit n Stufen [4].
- Varianzabhängige, kantenerhaltende Glättung auf beliebigen Stufen.
- Filterung über die letzten 3 Bilder (temporale Filterung).
- Zwischenspeicherung der gefilterten Hierarchie.
- Rekombination der Laplace-Pyramide zum Ergebnisbild.

Eigenschaften des implementierten Systems

- Framework für ein einfacheres Speichermanagement.
- Wiederverwendbarkeit und Erweiterungsmöglichkeiten durch ein modulares Filterdesign.
- Konfigurierbarkeit durch die Verwendung von Templates.

Optimierungen

- Verwendung des Texturcaches zur Minimierung der Latenz von Speicherzugriffen auf den globalen Grafikspeicher.
- Verwendung der effizienteren Fließkommaarithmetik.
- Verwendung der intrinsischen Funktionen (*–use_fast_math*)
  $\rightarrow$ effizienterer Code bei komplexen mathematischen Befehlen.
- Maximierung der Speicherbandbreite durch simultanes Lesen mehrerer Bildpunkte pro Thread (Datenbreite 16-bit, effizienteste Zugriffsbreite 32-bit), vgl. Abb. 2
  $\rightarrow$ verbessertes Alignment & günstigeres Verhältnis von Berechnungen zu Speicheroperationen.
- Erzwingung von automatischer Schleifenentrollung (loop unrolling) zur Übersetzungszeit durch Templates (der Compilerbefehl *#pragma unroll* funktioniert nicht immer).
- Schleifenentrollung ermöglicht das Entfernen einer Zugriffstabelle durch deren Speicherung im Code.
- Bestimmung der optimalen Grid-Konfiguration für jeden Einzelschritt.
- Reduktion divergenter Codepfade durch Minimierung von pfadinvarianten Anweisungen innerhalb der divergenten Pfade.

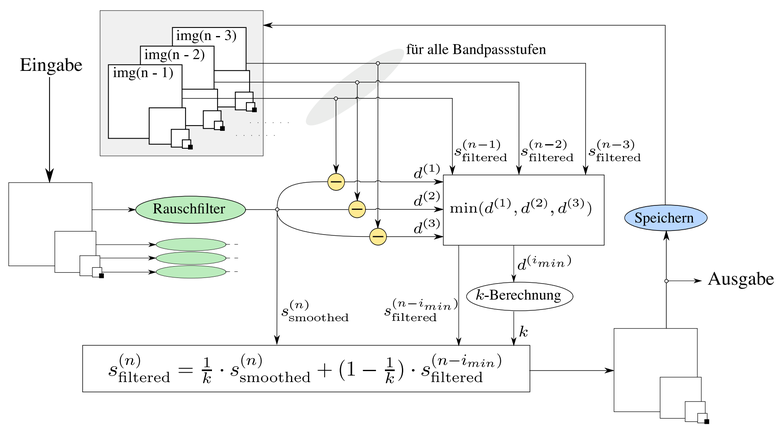

Abbildung 1: Verarbeitung der Bandpassbilder mit dem multiskalaren Rauschglättungsfilter; k-Faktor steuert den temporalen Einfluss.

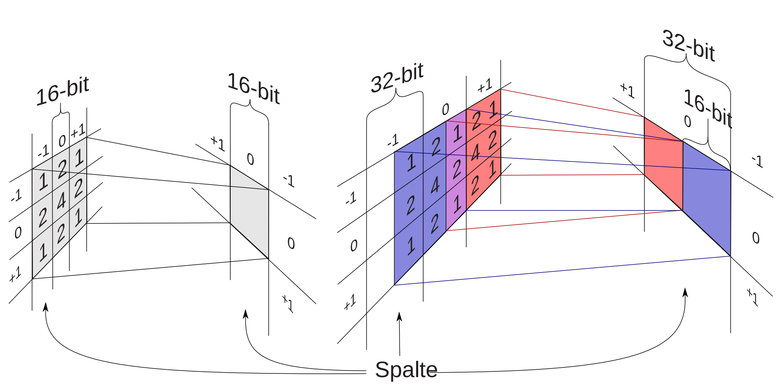

Abbildung 2: Sinnvolles Speicheralignment beim Downsampling mit einem Gausskern; links: Schreiben eines Wertes; rechts: Schreiben von zwei Werten innerhalb eines Kernels.

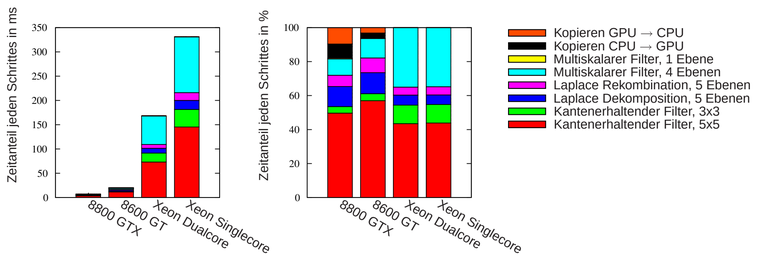

Abbildung 3: Rechenleistung des Multiskalaren Rauschglättungsfilters; Vergleich von GeForce 8800 GTX, 8600 GT, Singlecore XEON, Dualcore XEON; links: Zeiten in ms; rechts: Zeitanteil der einzelnen Filterschritte in %.

Experimentalaufbau

- Host System: Intel Xeon Prozessor (2.66 GHz), 4 GB Arbeitsspeicher
- GPU: NVIDIA GeForce 8800 GTX (345.6 Gflops, 128 Stream Prozessoren, 1.35 GHz, eine Multiply-add Operation pro Takt und Stream Prozessor)
- Beste Laufzeit aus 5 Messungen
- Datensatzgröße 960$\times$960 und 1024$\times$1024 Pixel, 12-bit Graustufen
- jeweils 20 - 40 Bilder

Evaluierung und Ergebnisse

- Durchschnittliche Abweichung von $7.2 \cdot 10^{-6}$ beim varianzgesteuerten Glättungsfilter, Wertebereich [0..4095], representatives Bild.
- Durchschnittlicher Fehler von etwa 2.78% beim multiskalaren Filter verursacht durch die rekursive Struktur.
- ca. $\times$45-fache Steigerung der Geschwindigkeit gegenüber der X86-Implementierung.
- Ziel 100ms Latenzzeit: weit übertroffen ($<$ 10ms).
- Erhöhte Bildqualität durch Fließkommaberechnungen (keine Auswirkungen durch intrinsische Funktionen).
- Unterschiedliches Leistungverhalten auf verschiedenen NVIDIA-Beschleunigerkarten, z.B. durch anderes Verhältnis von Rechenleistung zu Speicherbandbreite.

	Glättungsfilter	Konstruktion Laplace	Rekonstruktion Laplace	Multiskalarer Filter	Kopieren CPU→ GPU	Kopieren GPU→ CPU	Gesamt
Xeon 2.66 1 Kern	181.59	18.38	15.99	115.35			**331.31**
Xeon 2.66 2 Kerne	91.57	9.88	7.91	58.94			**168.30**
GeForce 8800 GTX	3.90	0.86	0.48	0.74	0.60	0.66	**7.24**
Speedup	$\times$46.6	$\times$21.4	$\times$33.3	$\times$155.9			$\times$**45.8**

Tabelle 1: Gemessene Zeiten jedes Filterschrittes in ms bzw. Beschleunigungsfaktor zwischen Singlecore und GPU Implementierung.

Zusammenfassung

- Architekturangepasste und allgemeine Optimierungsverfahren führen zu einer performanten CUDA-Implementierung eines komplexen 2D-Bildverarbeitungsalgorithmus.
- Beschleunigungsfaktor $\times$45 gegenüber X86 Singlecore Implementierung.
- Geschwindigkeitsvorteil mit Faktor 10 über den gegebenen Anforderungen.

Danksagung

Diese Arbeit wurde unterstützt von Siemens Healthcare, Imaging & IT, Angiography.

Literatur

[1] M. Mack, J. Hornegger, D. Paulus, A. Galant, and S. Böhm. Echtzeit-Röntgenbildverarbeitung mit Standardhardware. In T. Tolxdorff, J. Braun, Heinz Handels, A. Horsch, and H.-P. Meinzer, editors, *Bildverarbeitung für die Medizin 2004*, pages 395–399, Berlin, 2004.

[2] NVIDIA. *NVIDIA CUDA - Compute Unified Device Architecture - Programming Guide*. NVIDIA Corporation, 2701 San Tomas Expressway, Santa Clara, CA 95050 (USA), version 1.1 edition, November 2007.

[3] H. Scherl, B. Keck, M. Kowarschik, and J. Hornegger. Fast GPU-Based CT Reconstruction using the Common Unified Device Architecture (CUDA). In Eric C. Frey, editor, *Nuclear Science Symposium, Medical Imaging Conference 2007*, pages 4464–4466, 2007.

[4] P.J. Burt and E. H. Adelson. The laplacian pyramid as a compact image code. *IEEE Transactions on Communications*, 31:532–540, 1983.