MITK-OpenCL: Eine Erweiterung für das Medical Imaging Interaction Toolkit

Jan Hering[1,2], Ingmar Gergel[1], Susanne Krömker[2], Hans-Peter Meinzer[1], Ingmar Wegner[1]

[1]Abt. Medizinische und Biologische Informatik, DKFZ Heidelberg
[2]Visualisierung und Numerische Geometrie, IWR, Universität Heidelberg
jan.hering@iwr.uni-heidelberg.de

Kurzfassung. Die moderne medizinische Bildgebung ermöglicht immer detailliertere Daten, deren Weiterverarbeitung sich um so zeitaufwendiger gestaltet. Von der Medizin werden jedoch immer schnellere Bildverarbeitungsalgorithmen gefordert. Um diese Forderung zu erfüllen, müssen alle zur Verfügung stehenden Ressourcen genutzt werden. Die Grafikkarte ist eine dieser verfügbaren Ressourcen und kann für die Parallelisierung von Bildverarbeitungsalgorithmen herangezogen werden. Damit die Bildverarbeitungsprozesse unkompliziert auf die Graphikkarte ausgelagert und parallel berechnet werden können, wird in diesem Beitrag eine Hardware-unabhängige Erweiterung des Medical Imaging Interaction Toolkit vorgestellt. Die Ergebnisse zeigen eine wesentliche Beschleunigung der Algorithmen auf der Grafikkarte.

1 Einleitung

In der Bildverarbeitung kommen immer mehr rechenaufwendige Algorithmen zum Einsatz. Mit zunehmend hochauflösenden Datensätzen stellen diese Berechnungen einen Flaschenhals innerhalb einer Bildverarbeitungskettte dar. Die Chip Industrie reagiert auf diesen wachsenden Datendurchsatz mit der Entwicklung von parallelen Rechenarchitekturen. Hierbei handelt es sich einerseits um die multi-core Technologie für moderne Prozessoren (CPU) und andererseits um die many-core Technologie bei modernen Grafikkarten (GPUs). Der wesentliche Unterschied zwischen CPUs und GPUs liegt in der Verwendung des verfügbaren Platzes auf dem Chip. Da die CPU mannigfaltige Aufgaben bewältigen muss, widmet sie einen großen Teil des Chips der Logik und dem Cache. Die GPU dagegen stellt viele Berechnungseinheiten bereit, um so einen hohen Datendurchsatz zu gewährleisten. Folglich erreichen die neuesten GPUs eine Performanz von mehr als 1000 GFLOPS[1] im Gegensatz zu den CPUs mit nur 50 GFLOPS.

Durch einen erweiterten Zugang zu den Hardware Ressourcen der Grafikkarte für generelle Berechnungen (GPGPU [1]) werden GPUs immer häufiger in wissenschaftlichen Arbeiten eingesetzt. Um die Hardware-nahe Programmierung effizient und praktikabel zu machen, haben die GPU Hersteller NVIDIA

[1] 1 GFLOPS $= 1 \cdot 10^6$ floating point operations per second

und AMD eigene Programmiersprachen und Kompiler veröffentlicht (NVIDIA CUDA and ATI Stream). Diese jedoch sind nur auf die jeweilige Plattform begrenzt und bieten keine Portabilität. Aktuell steuert die GPU Programmierung in eine Hardware-unabhängige Zukunft. Am 8. Dezember 2008 erschien die erste Spezifikation von Open Common Language (OpenCL) Version 1.0, herausgegeben von der Khronos Group.

Um die GPU Beschleunigungsvorteile auch in der allgemeinen und speziell der medizinischen Bildverarbeitung auszunutzen, sind in letzter Zeit mehrere Projekte entstanden, die sich auf Parallelisierung von Algorithmen konzentriert haben. Die GpuCV Bibliothek [2], eine Erweiterung von Intel's OpenCV, implementiert einige Bildverarbeitung-Primitiven wie morphologische Operatoren und Kantendetektoren mithilfe von CUDA oder GLSL (OpenGL Shader Language). CUVILib konzentriert sich mehr auf Computer Vision Algorithmen, verwendet jedoch ebenfalls die plattformabgängige Sprache CUDA.

In diesem Beitrag wird eine Erweiterung für das Open Source Medical Imaging Interaction Toolkit (MITK [3]) vorgestellt, die durch eine OpenCL-Implementierung eine Hardware-unabhängige GPGPU Unterstütztung für die medizinische Bildverarbeitung bietet.

2 Methoden

Ziel der Erweiterung ist es, die Grafikkarte als allgemeine Recheneinheit für die Bildverarbeitung zur Verfügung zu stellen ohne dabei eine bestimmte Hardware-Konfiguration vorauszusetzen. Aus diesem Grund wurde für die Realisierung OpenCL gewählt. Stehen keine von OpenCL ansteuerbare Grafikkarten zur Verfügung, werden die Operationen wie gewohnt auf der CPU ausgeführt.

Voraussetzung für die Integration der Erweiterung war ein nahtloses Einfügen in die bestehenden Konzepte. Daher wurde auf ITK[2] Konzepten aufgebaut, welche sich in der medizinischen Bildverarbeitung als Standard etabliert haben. ITK ermöglicht es komplexe Berechnungen aus einzelnen Filtern in einer Bearbeitungspipeline zusammenzusetzen. Anhand der MITK OpenCL-Erweiterung lassen sich OpenCL-basierte Filter analog zu den ITK-Filtern in eine MITK Bearbeitungspipeline einreihen oder beliebig austauschen.

2.1 Filter- und Speicherverwaltung

Als erstes muss während der Laufzeit von einem Verwaltungsobjekt (contextmanager) untersucht werden, ob eine OpenCL-fähige Grafikkarte zur Verfügung steht und wenn ja, ob der Speicherbedarf des Filters auf der Grafikkarte gedeckt werden kann. Ist dies der Fall, so werden die Daten vom Filter in den Grafikspeicher kopiert. Jetzt können die parallelisierten Einzelberechnungen, die in einer dem Filter zugeordneten Datei definiert sind, mit hoher Datendurchsatzrate bearbeitet werden. Das Ergebnis der Berechnung wird ebenfalls auf der Grafikkarte

[2] Insight Toolkit

gespeichert. Wenn keine OpenCL-fähige Grafikkarte vorliegt oder nicht genügend Speicher frei ist, wird die Filter-Pipeline nicht unterbrochen, sondern die entsprechende CPU-basierte Version des Filters ausgeführt. Der context-manager ermöglicht weiterhin die filterübergreifende Verwendung eines einzelnen OpenCL-Kontextes, da ein ständiger Kontextwechsel unnötige Zeit beanspruchen würde. Zusätzlich können mehrere OpenCL-Filter durch die Verwaltung hintereinander angeordnet werden (Abb. 1). Um das Filterpipeline-Konzept zu realisieren und gleichzeitig die Latenz-Zeiten bei Speicherzugriffen minimal zu halten ist ein besonderer Umgang mit den Daten notwendig. Die zugehörigen Daten liegen entweder im Arbeitsspeicher, im Grafikspeicher oder sogar in beiden. Daher ist es vorteilhaft Daten erst dann aus dem GPU Speicher zu kopieren, wenn sie tatsächlich im Arbeitsspeicher gebraucht werden und vice versa.

Die MITK-OpenCL Erweiterung arbeitet mit eigenen Datenobjekten für Bild-, Oberflächen- und Punktmengendaten. Diese werden von den entsprechenden MITK Datenklassen abgeleitet und um die notwendigen Membervariablen und Methoden zum Verwalten der Kopiervorgänge zwischen RAM und dem Grafikspeicher erweitert.

2.2 Beispielanwendung

Neben üblichen Bildverarbeitungsalgorithmen (Konvolutionsfilter) wurde die Parallelisierung mit der Erweiterung auch für sequentielle Monte-Carlo Methoden (SMCM) durchgeführt. Diese Methoden arbeiten mit unabhängigen Zufallsproben (Partikeln), wobei die Genauigkeit einer SMCM direkt von der Anzahl der Partikel abhängt. Ein Einsatzgebiet für die SMCM ist das Kompensieren von Rauschen bei Navigationsaufgaben, wie zum Beispiel der navigierten Bronchoskopie [4].

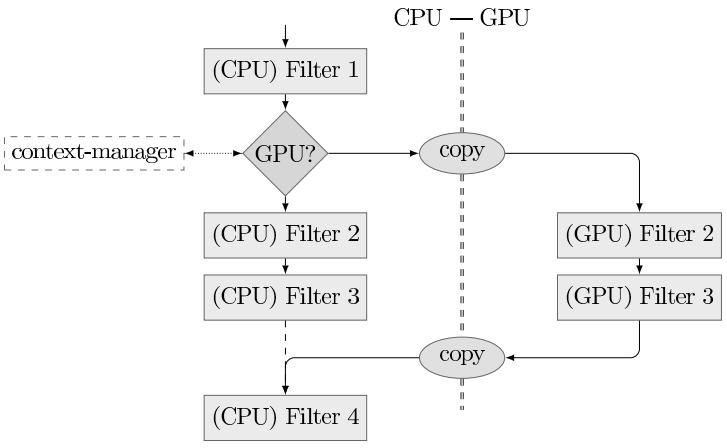

Abb. 1. Integration von OpenCL Filter in eine bestehende MITK Filter-Pipeline.

Tabelle 1. Laufzeitmessungen für einen Update-Schritt des Partikelfilters innerhalb der navigierten Bronchoskopie-Anwendung. Für die GPU Variante ist die Initialisierungszeit nicht inbegriffen, da sie nur einmalig beim ersten Update-Schritt nötig ist.

Partikel	128	256	512	1024	2048	4096	8192	16384
CPU [ms]	22.1	31.3	51.8	96.7	205.4	438.2	891.3	1712.1
GPU [ms]	9.2	7.3	8.9	9.0	9.2	10.9	12.8	17.9

3 Ergebnisse

Die Evaluation konzentriert sich auf die GPU Plattform. Die Laufzeitmessungen wurden mit folgender Hardwarekonfiguration durchgeführt – CPU: Intel Core i7 (Quadcore) 2.66 GHz; GPU: NVIDIA GeForce GTX 465. Gemessen wurde zuerst auf neun künstlichen Daten der Größe $2^k \times 2^l \times 2^m$ und anschließend auf zehn unterschiedlichen medizinischen Daten (Abb. 2).

Die Tabelle 1 fasst die erzielten Ergebnisse bei der parallelen Implementierung des Partikelfilters in der Beispielanwendung zusammen.

Die Laufzeitmessungen (Abb. 2) zeigen einen Beschleunigungsfaktor bei maximaler Bildgröße von 10 für den Gaussfilter bzw. 100 für den Opening-Filter. Die linke Grafik zeigt gleichzeitig ein typisches Phänomen für kleinere Daten-

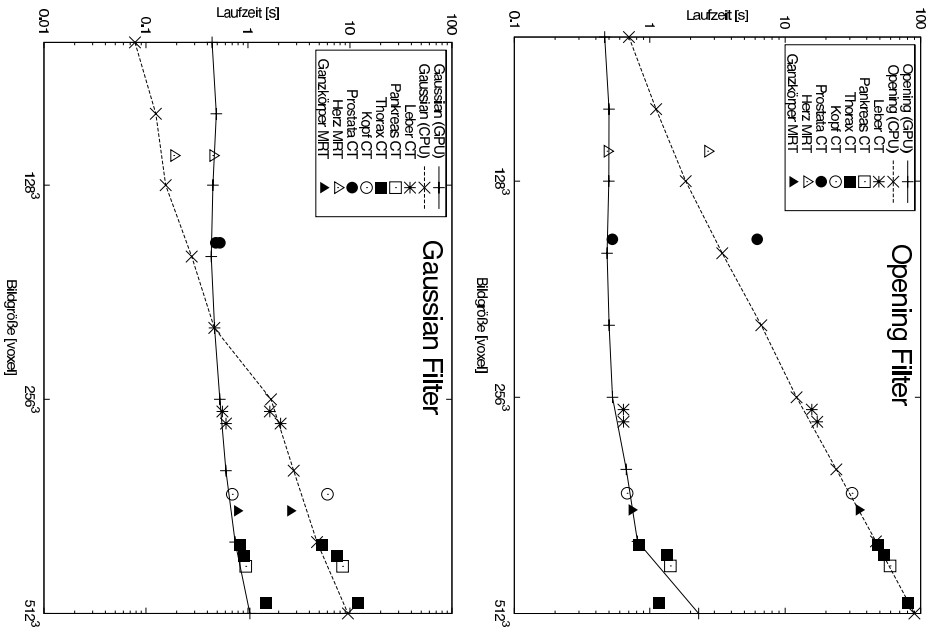

Abb. 2. Laufzeitmessungen für CPU bzw. GPU Implementierungen auf künstlichen Testdaten (+ bzw. ×) für Gauss- (links) und Opening-Filter (rechts) auf medizinischen Bilddaten verschiedener Modalitäten (CT, MR) und Scanbereiche.

sätze. Aufgrund der nötigen Initialisierungsschritte inklusive Kompilieren des Quellcodes für die GPU, muss man mit einer festen minimalen Laufzeit der GPU-basierten Implementierung rechnen. Diese liegt im Bereich von 300-400 ms. Jedoch wird dieser Zusatzbedarf nur bei der ersten Ausführung eines Filters benötigt. In der Beispielanwendung (Tab. 1) konnte durch die GPU Beschleunigung die Update-Rate des Partikelfilters deutlich erhöht werden, sodass auch für höhere Anzahl von Partikeln eine Anwendung in Echtzeit ermöglicht wird.

4 Diskussion

In der vorliegenden Arbeit wurde eine Hardware-unabhängige Erweiterung für GPGPU Anwendungen innerhalb von MITK vorgestellt. Die Erweiterung bietet eine Unterstützung für Entwickler, die nicht im Detail mit der GPU Programmierung vertraut sind und innerhalb ihrer Applikation die Parallelisierung auf der Grafikkarte für die Bildverarbeitung nutzen wollen. Die Erweiterung übernimmt für die Entwickler die OpenCL Initialisierung und Datenverwaltung, weiterhin können OpenCL-Filter analog zum Konzept der ITK-Filterpipeline beliebig eingegliedert werden.

Auch wenn ein direkter Laufzeitvergleich von CPU- und GPU-Implementierungen oft einen positiven Bias für die GPU besitzt [5], lohnt sich der Einsatz von OpenCL für neue Implementierungen, insbesondere für parallelisierbaren Code. Die Vertauschbarkeit innerhalb der ITK-Filterpipeline erleichtert eine Zusammenfügung mit bestehenden Filtern. Die vorgestellte MITK-OpenCL Erweiterung senkt das dafür nötige Einarbeitungspensum, somit kann mit wenig zusätzlichem Aufwand eine schnelle Implementierung fertiggestellt werden. Die MITK-OpenCL Erweiterung wird zur BVM2011 Open Source auf mitk.org zur Verfügung stehen.

Literaturverzeichnis

1. Luebke D, Harris M, Krüger J, et al. GPGPU: general purpose computation on graphics hardware. In: Proc SIGGRAPH; 2004. p. 33.
2. Allusse Y, Horain P, Agarwal A, et al. GpuCV: A GPU-accelerated framework for image processing and computer vision. Lect Notes Computer Sci. 2008;5359:430–9.
3. Wolf I, Nolden M, Böttger T, et al. The MITK approach. In: Proc MICCAI; 2005.
4. Gergel I, dos Santos TR, Tetzlaff R, et al. Particle filtering for respiratory motion compensation during navigated bronchoscopy. proc SPIE. 2010;7625(1):0W.
5. Lee VW, Kim C, Chhugani J, et al. Debunking the 100X GPU vs. CPU myth: an evaluation of throughput computing on CPU and GPU. SIGARCH Comput Archit News. 2010;38(3):451–60.