Vergleich Auralisierung über CPU und GPU

Michael Schöffler[1], Wolfgang Heß[1]

[1] *Harman/Becker Automotive Systems GmbH, 76307 Karlsbad, Deutschland, Email: michael.schoeffler@harman.com*

Einleitung

Die Auralisierung einer auditiven virtuellen Umgebung erfolgt, abhängig von der Kopfstellung, durch die Echtzeit-Faltung von Audiosignalen mit binauralen Raumimpulsantworten. Sie wird als plausibel wahrgenommen, wenn geringe Latenzen bei der Kompensation der Kopfbewegung erreicht werden, der virtuelle Raum also stabil ist.
Konventionelle Implementierungen basieren auf einer Berechnung der Auralisierung durch die CPU. Durch die Leistungsfähigkeit aktueller Grafikprozessoren kann die Berechnung teilweise oder komplett auf die Grafikkarte verlagert werden. Die Unterschiede hinsichtlich Umsetzung und Performance zwischen CPU und GPU werden anhand einer aktuellen Implementierung aufgezeigt.

Graphics Processing Unit

Die Graphics Processing Unit (GPU) ist der zentrale Prozessor von Grafikkarten. Bis vor wenigen Jahren wurde die GPU hautpsächlich für reine Grafikberechnungen zur Unterstützung der CPU verwendet. Die GPUs zeichnen sich gegenüber den CPUs hinsichtlich der großen Anzahl an gleichzeitig nutzbaren Rechenkernen aus. Dadurch können Berechnungen, die sich in parallele Teilberechnungen zerlegen lassen, deutlich schneller durchgeführt werden.

Durch den einfachen Zugriff auf die GPU mit Hilfe von Programmierplattformen, wie CUDA[2] und OpenCL[3], ist es möglich, diese für die Verarbeitung von nicht-grafikbezogenen Daten zu verwenden. OpenCL hat gegenüber CUDA den Vorteil, dass es von der Khronos Group standardisiert wurde und von unterschiedlichen Grafikkartenherstellern unterstützt wird. CUDA hingegen kann ausschließlich mit NVIDIA-Karten verwendet werden. In einem direkten Vergleich zwischen CUDA und OpenCL zeigte sich, dass die Ausführungszeit von OpenCL-Programmen um 13% bis 86% langsamer ist[4]. Trotzdem wurde im Rahmen der Arbeit OpenCL verwendet um den Vorteil, sich nicht auf eine bestimmte Herstellerplattform festlegen zu müssen, zu nutzen.

Faltungsalgorithmen

In der Vergangenheit wurden mehrere Faltungsverfahren publiziert, die alle unterschiedliche Vor- und Nachteile mit sich bringen. Neben einer gewöhnlichen Faltung im Zeitbereich ermöglicht es das Faltungstheorem, die Faltung im Frequenzbereich durch eine komplexe Multiplikation von Eingangsaudiodaten und Raumimpulsantworten ab einer bestimmten Operandengröße wesentlich schneller durchzuführen. Diese als Schnelle Faltung bekannte Methode ist insbesondere in der Variante des Overlap-Save-Verfahrens [1] von Vorteil, da sich die Gesamtberechnung des Ausgabestroms in mehrere unabhängige Teilberechnungen einteilen lässt und diese auf heutigen CPUs und GPUs parallel ausgeführt werden können.

Methode

Der Vergleich zwischen CPU- und GPU-Faltungen erfolgt in dieser Arbeit anhand von vier Implementierungsansätzen, drei davon im Frequenzbereich und einer im Zeitbereich.

Da beim Gebrauch einer GPU andere Randbedingungen und zusätzliche Latenzen, wie der Datentransfer vom Hauptspeicher zur Grafikkarte, beachtet werden müssen, stellt sich die Frage, welche Teilberechnungen der Faltung durch die CPU und durch die GPU ausgeführt werden. In [5] wurde ein Ansatz vorgestellt, bei dem die Schnelle Fourier-Transformation durch die CPU und die Multiplikation von Eingabestrom und binauralen Raumimpulsantworten auf der GPU berechnet werden.

In dieser Arbeit werden folgenden vier Varianten miteinander verglichen:

CPU Alle Berechnungen werden durch die CPU ausgeführt.

CPU/GPU Die Schnellen Fourier-Transformationen zur Überführung in den Frequenzbereich werden auf der CPU ausgeführt. Die notwendigen Multiplikationen und Summierungen werden vollständig auf der GPU gerechnet.

GPU Alle relevanten Berechnungen werden durch die GPU ausgeführt. Die CPU steuert lediglich den Transfer des Ein- und Ausgabedatenstrom zu und von der GPU.

GPU im Zeitbereich Eine Faltung im Zeitbereich gewinnt ab einer bestimmten Eingangsgröße sehr schnell an Rechenaufwand, da diese bei einer gewöhnlichen Umsetzung quadratisch zunimmt. Dieses Variante wurde miteinbezogen, um zu prüfen, in wie weit sich diese Eigenschaft auf die Gesamtberechnungszeit der GPU auswirkt.

Die vier unterschiedlichen Varianten sind in eine Faltungsmaschine zur Erzeugung einer auditiven virtuellen Umgebung integriert. Die Faltungsmaschine berechnet

eine frei wählbare Anzahl von Eingangskanälen mit beliebigen Filterdatensätzen. Die Filterdatensätzen enthalten binaurale Raumimpulsantworten unterschiedlicher Kopfstellungen. Die passende Raumimpulsantwort für eine Faltung wird durch Tracking des Hörerkopfes bestimmt. Ändert sich die Kopfstellung, werden jeweils die letzte und aktuelle Raumimpulsantwort mit dem Eingangsstrom gefaltet und anschließend ineinander überblendet (Cross-Fading).

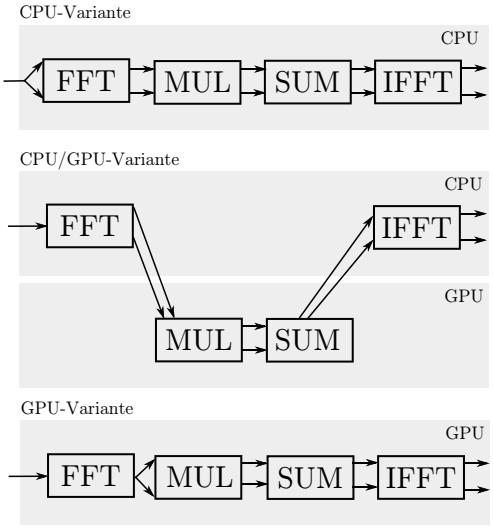

Abbildung 1: Schematische Darstellung der drei Varianten, die die Berechnung im Frequenzbereich durchführen.

Bei allen Verfahren wurden alle Befehle, die auf der CPU ausgeführt wurden, soweit wie möglich mit der IPP-Bibliothek[6] von Intel umgesetzt. Diese Bibliothek nutzt spezielle Prozessorbefehle, wodurch Geschwindigkeitsvorteile gegenüber einer konventionellen Befehlsnutzung erzielt werden. Bei dem vollständigen GPU-Ansatz wurde für die Schnelle Fourier-Transformation die FFT-Implementierung[7] von Apple verwendet.

Auswertung

Alle Implementierungsansätze wurden mit sechs Eingangskanälen bei einer Blockgröße von 256 Samples mit unterschiedlich langen binauralen Raumimpulsantworten gefaltet. Bei der Messung wird für jeden Faltungsvorgang von einer festen Kopfstellung ausgegangen. Die Ergebnisse zeigen die durchschnittliche Berechnungszeit, die für einen kompletten Faltungsvorgang benötigt wurde. Da eine Abtastrate von 48000 Hz und eine Blocklänge von 256 Samples verwendet wurde, liegt die kritische Grenze für eine Echtzeitfaltung bei $\frac{256[samples]}{48000[samples/s]} = 5.3ms$ maximaler Berechnungszeit.

Abbildung 2 zeigt das erzielte Resultat. Es wird deutlich, dass die CPU-Variante für kurze Raumimpulsantworten deutlich schneller ist als die anderen Varianten. Mit zunehmender Länge der Raumimpulsantworten nimmt der Geschwindigkeitsvorteil der beiden GPU-Varianten im Frequenzbereich zu. Dies ist damit begründbar, dass bei jeder Faltung auf der GPU Mehraufwände, wie Datentransfers oder Befehlssteuerungen zur GPU, entstehen. Diese Mehraufwände werden bei längeren Raumimpulsantworten durch die stark parallele Berechnungsmöglichkeit der GPU ausgeglichen. Die Faltung im Zeitbereich auf der GPU ist aufgrund der quadratischen Zunahme des Rechenaufwands gegenüber den anderen Varianten deutlich langsamer. Es zeigt sich, dass bei Messungen mit den größten Filterlängen (6 Eingangskanäle mit 262144 Samples langen Binauralfiltern), die reine GPU-Variante mit durchschnittlich $3.09ms$ deutlich schneller ist als die reine CPU-Variante mit $6,35ms$ und auch schneller wie die gemischte CPU/GPU-Variante mit $3,50ms$.

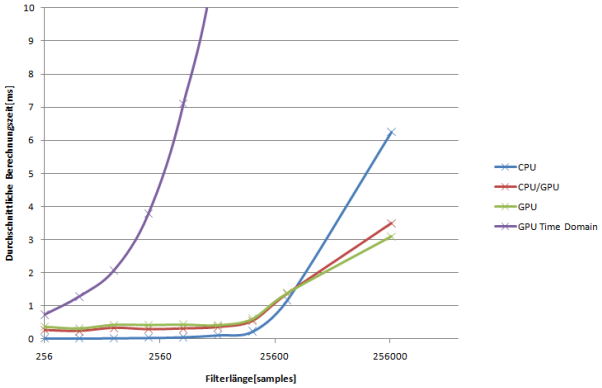

Abbildung 2: Durchschnittliche Berechnungsdauer einer Faltung der verschiedenen Verfahren auf einem Intel Core i7 860 mit einer NVIDIA GTX 470 Grafikkarte.

Literatur

[1] Kulp: Digital Equalization Using Fourier Transform Techniques, 85th AES Convention Preprint, Los Angeles, 1988

[2] Compute Unified Device Architecture, URL: `http://developer.nvidia.com/object/gpucomputing.html`

[3] OpenCL, Offene Programmierplattform für den Zugriff auf GPUs, URL: `http://www.khronos.org/opencl/`

[4] Karimi, Dickson, Hamze: A Performance Comparison of CUDA and OpenCL, arXiv.org, 2010

[5] Wefers, Berg: High-Performance Real-Time FIR-Filtering Using Fast Convolution on Graphics Hardware, Proc. of the 13th Int. Conference on Digital Audio Effects (DAFx-10), Graz, 2010

[6] Integrated Performance Primitives, URL: `http://software.intel.com/en-us/articles/intel-ipp/`

[7] Apple OpenCL FFT, OpenCL-Implementierung einer FFT, URL: `https://developer.apple.com/library/mac/#samplecode/OpenCL_FFT/Introduction/Intro.html`