Echtzeitfähiges binaurales Rendering mit Bewegungssensoren von 3D-Brillen

Hagen Jaeger[1], Joerg Bitzer[1], Uwe Simmer[1] and Matthias Blau[1]
[1] *Institut für Hörtechnik und Audiologie, Jade Hochschule Oldenburg*

Kurzbeschreibung

Die Möglichkeit zur Erzeugung virtueller Realität ist im Endverbrauchermarkt angekommen. Aufgabe der Akustik ist es, realistische Hörsituationen durch geeignete Signalverarbeitung zu simulieren. Moderne Consumer-Hardware, wie etwa eine 3D-Brille oder ein Smartphone, bietet Zugang zu qualitativ hochwertigen Lagesensoren, die genutzt werden können, um Echtzeit-Headtracking zu realisieren. Durch Einsatz einer zeitvarianten, latenzarmen Filterung von Kopfübertragungsfunktionen kann eine natürliche Richtungswahrnehmung bei variierenden Kopfausrichtungen gewährleistet werden. Zusätzlich dazu können räumliche Informationen durch geeignete Verarbeitung von Mehrkanalaufnahmen eines virtuellen Kunstkopfes individualisiert gerendert werden [2], [3], [4]. Ein entsprechend ausgelegtes Softwaredesign erlaubt die Erstellung einer Plattformübergreifenden Lösung, die auf standardisierten Datenstrukturen arbeitet.

Einleitung

Die Anwendung von Freifeld-Kopfübertragungsfunktionen und binauralen Raumimpulsantworten ist eine gängige Praxis in der virtuellen Akustik, um Schallquellenpositionen und räumliche Höreindrücke zu simulieren.

Rasumow et al. (2014-2016) stellten in ihren Arbeiten die Anwendung eines virtuellen Kunstkopfes zur Messung räumlicher Informationen vor. Durch passende Signalverarbeitungsstrategien der resultierenden Mehrkanalaufnahmen können damit einmalig gemessene räumliche Schallfelder mit individuellen Kopfübertragungsfunktionen wiedergegeben werden [2], [3], [4].

Bei der Echtzeit-Auralisation ist die Aufgabe der Signalverarbeitung, eine perzeptiv artefaktfreie Filterung mit möglichst geringer Latenz zu ermöglichen. Wefers (2014) stellt in seiner Arbeit über partitionierte Faltungsalgorithmen zur Echtzeit-Auralisation verschiedene Methoden vor, die durch Partitionierung und Frequenzbereichsverarbeitung latenzarm, sowie effizient arbeiten [1]. Für Signale mit vornehmlich hochtonaler Gewichtung konnte eine effektive Maskierung von Artefakten bei zeitvarianten Filtervorgängen beobachtet werden, wohingegen überwiegend tieffrequent energetische Signale zu wahrnehmbaren Störgeräuschen in dieser Situation führten.

Durch Modifikation der partitionierten Faltung nach dem Überlappungsprinzip konnten im Rahmen der vorliegenden Arbeit die Anforderung der perzeptiven Artefaktfreiheit für zeitvariante Filtervorgänge erfüllt werden. Die folgenden Abschnitte beschäftigen sich mit der Software-Implementierung und den Ergebnissen der Arbeit.

Implementierung

Um echtzeitfähiges binaurales Rendering zu realisieren, wird ein Mehrkanal-Datenstrom aus Audiosamples durch eine Filterroutine bearbeitet, deren Übertragungsverhalten sich je nach Kopfausrichtung so ändert, dass eine passende Kopfübertragungsfunktion auf das Eingangssignal, bzw. die Eingangssignale, angewendet wird.

Eine direkte Berechnung mittels blockorientierter, übelappender Faltung im Frequenzbereich führt zu wahrnehmbaren Artefakten bei zeitvarienten Filtervorgängen. Diese äußern sich als Knackgeräusche bei der Umschaltung von Filterübertragungsfunktionen. Durch Modifikation der partitionierten Faltung im Frequenzbereich war es möglich, diese Störgeräusche zu unterdrücken.

Für jeden Kanal des Datenstroms wurden zu 50% überlappende Fenster der Länge L zu den Analysezeitpunkten K benutzt, um eine Einteilung in Signalblöcke vorzunehmen. Die Fensterfunktionen werden als periodische, verschobene Hann-Fenster $w_{\text{hann}}(k-K)$ definiert und durch Anfügen von L Nullen auf die doppelte Blocklänge $2L$ erweitert (Durch ein hochgestelltes z am Ergebnis notiert). Nachfolgend wird eine Fouriertransformation berechnet, sodass

$$X_w^z(\Omega, K) = \sum_{k=K}^{K+2L-1} x(k) \cdot w_{\text{hann}}(k-K) \cdot e^{-j2\pi \frac{k}{2L}} \tag{1}$$

für alle diskreten Frequenzen Ω gilt. Um eine partitionierte Faltung im Frequenzbereich zu realisieren werden die Impulsantworten für jede Kopfausrichtung ($h_{\theta,\Theta,\Psi}(k)$) in N Partitionen der Länge L unterteilt, welche ebenfalls nach einem Zeropadding auf die doppelte Blocklänge in den Frequenzbereich transformiert werden. Dabei bezeichnen θ, Θ, Ψ den Azimuth-, Elevations- und Kippwinkel (engl. „Yaw, Pitch, Roll“), zu denen die Impulsantwort bestimmt wurde (Speicherung als Metadaten zur Messung). Es folgt

$$H^z_{\theta,\Theta,\Psi}(N,\Omega) = \sum_{k=NL}^{(N+2)L-1} h_{\theta,\Theta,\Psi}(k) \cdot w_{\text{rect}}(k-L) \cdot e^{-j2\pi\frac{k}{2L}} \tag{2}$$

als Repräsentation für alle im Frequenzbereich partitionierten Impulsantworten. Durch Multiplikation aller N vergangenen Signalblöcke mit ihren zugehörigen Filterpartitionen kann das Ausgangssignal im Frequenzbereich als

$$Y^z_w(\Omega,K) = \sum_{n=0}^{N} X^z_w(\Omega,K-n) \times H^z(n,\Omega) \tag{3}$$

beschrieben werden. dabei is n Element der geraden, natürlichen Zahlen $[0,2,4,6,...]$ um ausschließlich nicht überlappende Faltungsergebnisse miteinander zu verrechnen und $\times$ bezeichnet die elementweise Vektormultiplikation. $H^z(N,\Omega)$ wird durch die aktuelle Kopfausrichtung des Zuhörers determiniert, indem ein Vergleich zu den Metadaten der Impulsantworten angewandt wird. Ein Suchalgorithmus findet jene Impulsantwort, deren Messwinkel die geringste euklidische Distanz zu den aktuellen Werten der Kopfausrichtung aufweisen. Diese wird aus Sensordaten als $\theta(k), \Theta(k)$ und $\Psi(k)$ bestimmt,

$$H^z(N,\Omega) = H^z_{\Phi,\theta,\Theta}(N,\Omega) \text{ mit } \Phi(k), \theta(k), \Theta(k). \tag{4}$$

Der Algorithmus kann mittels Extremwertanalyse auf den Winkeldifferenzen von Metadaten und Kopfausrichtungen arbeiten, das absolute Minimum charakterisiert die beste Anpassung. Mithilfe der inversen Fouriertransformation und einer überlappenden Addierung von Filterteilergebnissen kann das Zeitbereichs-Ausgangssignal durch

$$y(k) = \sum_{K=0}^{\infty} \left(\sum_{\Omega=0}^{2L} Y^z_w(\Omega,K) \cdot e^{j2\pi\frac{\Omega}{2L}} \right) \tag{5}$$

bestimmt werden. Eine Visualisierung des schematischen Ablaufes ist im blauen (Audiosignalverarbeitung), bzw. im grünen (Impulsantwortverarbeitung) Strang des Blockschaltbildes in Abbildung 1 zu finden.

Die Zuführung der Winkeldaten wurde per Netzwerk im „User Data Protocol“ (UDP), bzw. durch serielle Übertragung per „Universal Serial Bus“ (USB) realisiert. Der Suchalgorithmus und die Zuführung der Kopfausrichtung sind im genannten Blockschaltbild als roter Strang, bzw. gelber Knotenpunkt visualisiert.

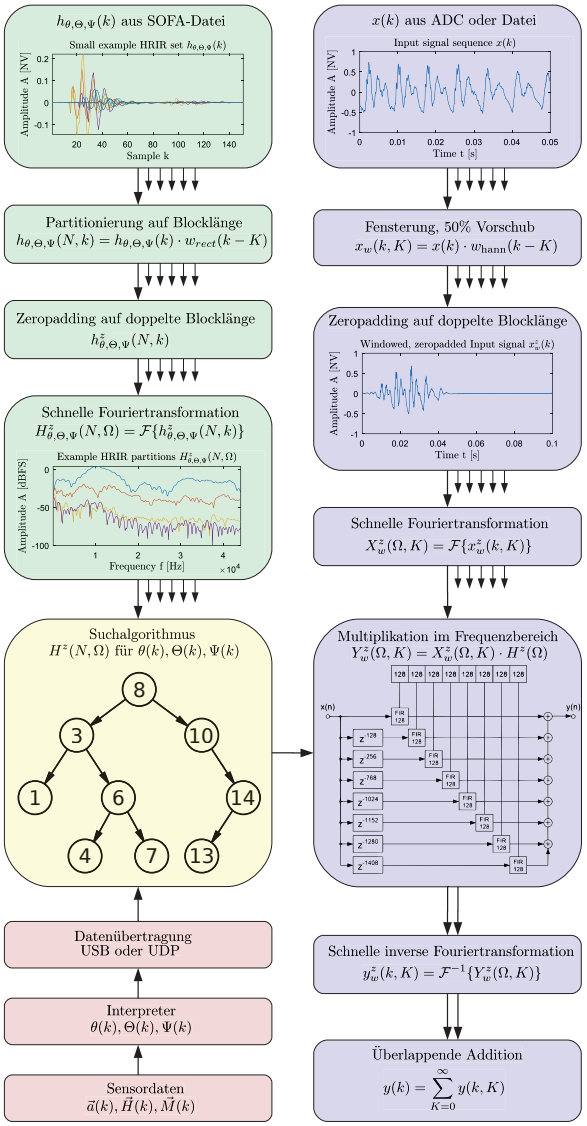

Abbildung 1: Ablaufdiagramm des binauralen Renderers (Blockschaltbild unter „Multiplikation im Frequenzbereich“ aus [1])

Für den Fall der direkten Anwendung von Kopfübertragungsfunktionen zur Simulation von Quellpositionen kann eine Mono-Audiodatei eingespeist und für das linke und rechte Ohr die per Suchalgorithmus gefundene Kopfübertragungsposition aufgeprägt werden. Im Fall einer mehrkanaligen Aufnahme mittels

Kunstkopf kann jeder Kanal mit den entsprechenden Filterkoeffizienten für das jeweilige Ohr multipliziert, und alle Ergebnisvektoren summiert werden [3]. Die durch eine partitionierte Faltung aufgeprägte Verzögerung beträgt L Audiosamples.

Ergebnisse

Die Richtigkeit der im vorherigen Kapitel vorgestellten Echtzeit-Faltungsoperation wurde mithilfe der Filterung einer Rechteckschwingung mit einer verzögerten, abklingenden und invertierten Kosinusfunktion überprüft (Abbildung 2). Die unterste Grafik zeigt das Ergebnis der modifizierten partitionierten Faltung nach dem Überlappungsprinzip. Im Vergleich dazu zeigt die darüberliegende Grafik das Ergebnis mittels direkter Zeitbereichsfaltung über die Verschiebung κ als

$$y(k) = \sum_{\kappa=-\infty}^{\infty} x(k) \cdot h(\kappa - k) = x(k) * h(k). \qquad (6)$$

Es ist zu erkennen, dass die implementierte Faltungsoperation eine zusätzliche Verzögerung erzeugt, jedoch ansonsten visuell identische Ergebnisse liefert. Die zusätzliche Verzögerung entspricht in diesem Fall der Blocklänge $L = 1024$ Samples, beziehungsweise $\approx 21,3$ms mit $f_{\text{samp}} = 48$kHz. Wird die Blocklänge L auf 128 Samples gesetzt ergibt sich eine Verzögerung von $\approx 2,67$ms, die gering genug ist um nicht hörbar zu sein.

Da die partitionierte Faltung nach dem Überlappungsprinzip eine Effizienzsteigerung gegenüber der direkten Zeitbereichsfaltung bewirkt [1], kann neben der perzeptiven Artefaktfreiheit auch eine Verringerung des Rechenaufwandes als Ergebnis angesehen werden. Die Größe der Effizienzsteigerung ist abhängig von der gewählten Blocklänge L und um etwa den Faktor Zwei geringer als bei der nicht modifizierten partitionierten Faltung nach dem Überlappungsprinzip. Dieser Tatsache liegt der Vorschub von 50 statt 100% Blocklänge und die zusätzliche Fensterung zugrunde.

Das Einlesen von Audiodaten wurde mithilfe der Softwarebibliotheken „libmpg123“ und „libsndfile“ realisiert [6], [7]. Dadurch ist es möglich, sowohl nach dem MPEG-Layer 3 verlustbehaftet, als auch verlustfrei komprimierte oder unkomprimierte Audiodateien einzulesen. Das Zuführen von Filterdaten für Kopfübertragungsfunktionen wurde mit Hife des SOFA-API als „SimpleFreeFieldHRIR“-

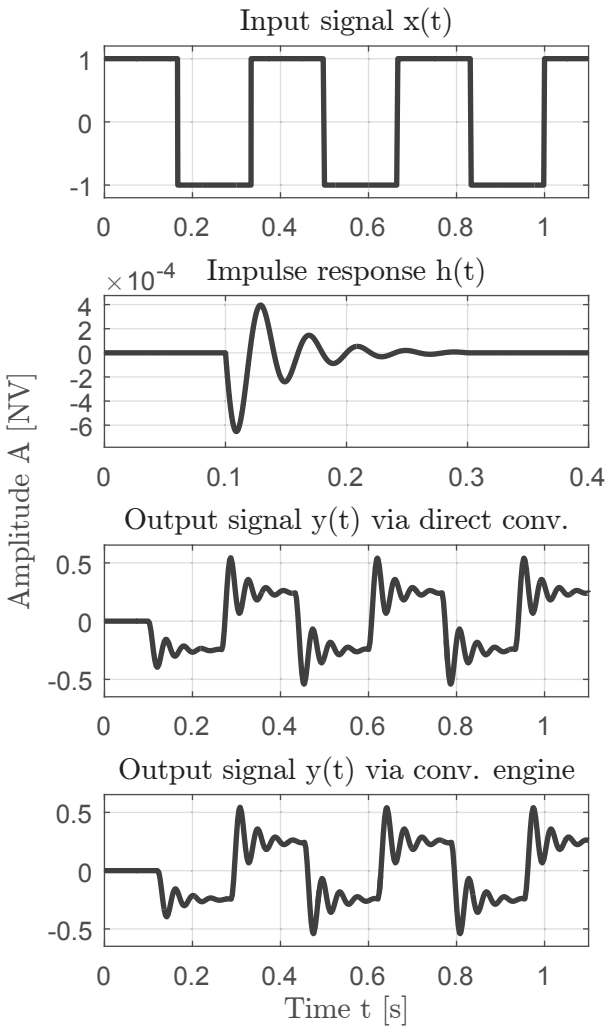

Abbildung 2: Test der modifizierten partitionierten Faltung nach dem Überlappungsprinzip durch Vergleich zu einer direkten Faltung im Zeitbereich

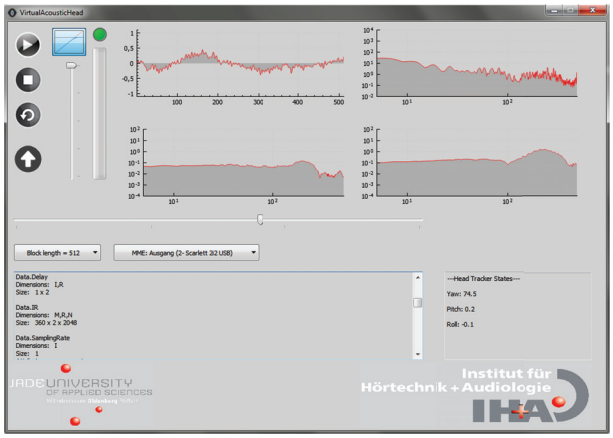

Abbildung 3: Grafische Benutzeroberfläche des binauralen Renderers

Konvention realisiert [5]. Dadurch ist eine transparente Zuführung von Datenströmen an das Programm realisiert worden. Zudem wurde mithilfe des Qt-API eine grafische Benutzeroberfläche imple-

mentiert, die Ausführung und Bedienung des Programms vereinfacht [9]. Eine Version der aktuellen Benutzeroberfläche unter Windows ist in Abbildung 3 dargestellt. Die Zuführung von Sensordaten per Netzwerk (UDP) oder über eine USB-Schnittstelle wurde mithilfe von Qt-Bibliotheken und dem OVR-API zur Einbindung der Oculus Rift DK2 realisiert [8]. Dadurch werden eingehende Sensordaten der „FreePIE IMU Sender"-Smartphone-App als Teil von Opentrack [10], sowie Datenströme des Polhemus Fastrak Pro und der Oculus Rift DK2 standardmäßig unterstützt.

Da alle Teilkomponenten der Software nach dem C++-Standard 2011 implementiert sind und bei der Bibliothekenauswahl Wert auf plattformübergreifende Lösungen gelegt wurde, ist die hier vorgestellte Software generell für alle gängigen Betriebssysteme (Apple OSX, MS Windows und Linux) erstellbar. Die Software ist momentan ohne Multicore-Support implementiert. Zur Berechnung eines zehnsekündigen Faltungshalls mit Blocklänge $L = 512$ und zwei Kanälen ($\widehat{=}$ 1876 Faltungspartitionen) benötigt die Ausführung etwa 50% der Ressourcen eines einzelnen Rechenkerns auf einem Intel i5 Prozessor der 6. Generation (Standard Consumer-PC-Hardware).

Literatur

[1] Wefers, F. (**2014**): „Partitioned convolution algorithms for real-time auralization". Dissertation RWTH Aachen

[2] Rasumow E., Blau M., Hansen M., van de Par S., Doclo S., Mellert V., Püschel D. (**2014**). *J. Acoust. Soc. Am.*, **135**(4): 2012–2025.

[3] Rasumow E., Blau M., Doclo S., Hansen M., van de Par S., Püschel D., Mellert V. (**2015**). *Fortschritte der Akustik - DAGA 2015*, Nürnberg.

[4] Rasumow E., Hansen M., van de Par S., Püschel D., Mellert V., Doclo S., Blau M. (**2016**). *IEEE/ACM Transactions in Audio, Signal and Language Processing*, **24**(4): 215–225.

[5] AES 69-2015 (**2015**): „AES standard for file exchange - Spatial acoustic data file format". Audio Engineering Society Inc. 551 Fifth Avenue, New York.

[6] libmpg123 software library, URL: `https://www.mpg123.de/`

[7] libsndfile software library, URL: `http://www.mega-nerd.com/libsndfile/`

[8] Oculus virtual reality software library, URL: `https://developer.oculus.com/`

[9] Qt5 software library, URL: `https://www.qt.io/`

[10] Opentrack: Head tracking software for MS Windows, Linux, and Apple OSX `https://github.com/opentrack/`