EveryDrum

Autoren:Tom Ziegler[1], **Lukas Peschel**[1]
Betreuer: Dr. Sebastian Merchel[1]

[1]Technische Universität Dresden

Erstellt am 9. Juli, 2018

Abstract

Drums | Android | Arduino | PureData |

Einleitung

Diese Projektarbeit entstand im Rahmen des Modules "Virtuelle Realitätïm Sommersemeter 2018. Die Zielstellung war die Entwicklung eines Systems, bestehend aus Hard- und Software, mithilfe dessen eine beliebige Oberfläche als Schlagzeug verwendet werden kann. Eine angeschlagene Position soll dafür in Echtzeit klasifiziert und einem der vorher erlernten Schlaginstrumente zugeordnet werden.

Die Vergabe der Projektthemen und erste Besprechung dieser erfolgte am 16.04.2018. Am 28.05. erfolgte eine Zwischenpräsentation, im Rahmen dieser die bisherigen Konzepte der verschiedenen Teams gegenseitig vorgestellt und diskutiert wurden. Die Abgabe und abschließende Präsentation der Ergebnisse erfolgte am 09.07.2018. Der Zeitraum der Bearbeitung erstreckte sich damit über 12 Wochen.

Im Rahmen einer ersten konzeption wurden folgende Maßgaben für die Umsetzung aufgestellt:

- Das entstandene System soll portabel, einfach anzuwenden und wenig spezielle Hardware erfordern
- Um ein realistisches Spielgefühl zu erzeugen, soll die maximale Latenz zwischen Anschlag und Abspielen des Sounds 100ms betragen
- Das System soll robust funktionieren und dem Anwender ein Feedback über die Zuverlässigkeit der Erkennung liefern, sodass dieser eventuell Verbesserungsmaßnahmen durchführen kann (z.Bsp. Reduzierung von Hintergrundrauschen)

Um die Realisierung der Projektidee in diesem zeitrahmen möglich zu machen, wurden zu Beginn einige Vereinbarungen getroffen:

- Als Oberfläche sollen vorerst nur Tische mit Holz- oder Kunststoffplatten verwendet werden
- Maximal sollen drei verschiedene Positionen unterschieden werden
- Im Sinne einer Prototypentwicklung sollen Funktionalität und Beweis der Konzeptplausibilität höhere Priorität als Benutzerfreundlichkeit haben
- In diesem Projekt soll die Eigenschwingung der angeschlagenen Oberfläche zur Auswertung genutzt werden.

Fragestellungen

- Welche Oberflächen eignen sich am besten?
- Wie groß muss das Zeitfenster für einen Schlag gewählt werden, um eine robuste Erkennung sowie eine geringe Latenz zu erreichen?
- Welche Informationen könne aus den Beschleunigungsdaten gewonnen werden, und welche eignen sich zur Echtzeit-Klassifizierung?
- Wie kann die Überlagerung zweier Anschläge verhindert/verringert werden?

1. Konzept

Ein naiver Ansatz wäre, mehrere Sensoren auf der Oberfläche zu verteilen und eine ein- oder zweidimensionale Lokalisation mittels Bestimmung der Laufzeitunterschiede durchzuführen. Dies setzt allerdings die Verwendung mindestens zweier (eindimensional, max. 3 Positionen) bzw. dreier (zweidimensional) Sensoren. Die Verwendung mehrerer Sensoren bedeutet jedoch auch steigende Anforderungen an die verarbeitende Hardware, höherer Realisierungsaufwand, höhere Anforderungen an den Anwender und nicht zuletzt höheren Hardwareaufwand und -abhängigkeit, was die Portabilität und Verwendbarbarkeit deutlich einschränkt.

Daher soll lediglich ein Sensor verwendet werden. Dies bedeutet jedoch, dass die Unterscheidung der Anschlagpositionen lediglich anhand der Schwingungscharakteristiken erfolgen muss.

Wird eine Tischplatte an einer Position angeschlagen, führt diese eine Eigenschwingung aus. Je nach Material, Lagerung, Obejekte auf der Oberfläche, etc... unterscheiden sich verschiedene Tischplatten in ihrer Impulsantwort. Außerdem unterscheidet sich die ausgeführte Schwingung je nach Position und Art der Erregung: Ein Schlag mit den Fingerknochen erzeugt eine andere Schwingung als ein Schlag mit der flachen Hand - dies lässt sich leicht anhand des entstehenden Geräusches verifizieren. Ebenso erzeugt ein Schlag auf die Tischkante eine andere Schwingung als ein Schlag in die Tischmitte - dies zeigt sich vor allem in unterschiedlichen Nachhallzeiten.

Gewisse charakteristische Eigenschaften sind bauartbedingt durch den Tisch vorgegeben. Diese können vom Anwender nicht beeinflusst werden, folglich muss die Software in der Lage sein, diese Unterschiede in der Verarbeitung zu berücksichtigen. Trotzdem hat der Anwender gewisse Möglichkeiten, die Schwingung der Tischplatte zu beeinflussen, beispielsweise durch Anbringen von Massen, Dämpfern, etc... Dies kann und soll bewusst eingesetzt werden, um die Schwingungscharakteristiken der Anschlagspositionen möglichst unterschiedlich zu gestalten.

Um eine hohe Flexibilität in der Anwendung zu liefern, soll auf die Verwenung eines PCs/Laptops verzichtet werden. Stattdessen soll die softwareseitige Umsetzung mittels eines Smartphones erfolgen.

Zur Aufnahme der Schwingung wurden verschiedene Möglichkeiten erprobt:

- Verwendung eines Piezzosensors
- Verwendung des internen Beschleunigungssensors des Smartphones
- Verwenung eines Arduino Mikrocontrollers mit Beschleunigungssensoren

2. Umsetzung

Für die Umsetzung wurde ein Android-Smartphone, ein Lenovo K6, mit der Android-Version 7.0 verwendet. Hingegen zu älteren Smartphones unterstützen neuere den USB On-The-Go (OTG) Modus, womit die Stromversorgung des Arduinos über USB-Anschluss gewährleistet wird.

Desweiteren wurde ein Arduino CH340 chip verwendet und als Beschleunigungssensoren das Modell MPU-6050. Die Android-App wurde in Java geschrieben. Zur Kommunikation mit dem Arduino und abgreifen der Beschleunigungssensoren des Sensors wurden die Bibliotheken *usb-serial-for-android* [mwm18] und *UsbSerial* [Her18] verwendet.

Die synthetisch erzeugten Schlagzeuggeräusche für Hihat, Snare und Bass wurden mit PureData [uIoEMA18] erzeugt. Zur Nutzung der PureData-Patches auf dem Android Gerät, wurde die Bibliothek *pd-for-android* [lu18] von *libpd* verwendet. Diese unterstützt keine erweiterten Funktionen, wie sie in PureData Extended vorkommen.

3. Arduino

Arduino Mikrocontroller

Für die Umsetzung wurde ein Ärduino Nano"Mikrocontroller mit Prozessor ATmega328P verwendet. Dieser führt selbst keine Datenverarbeitung aus, sondern bietet lediglich die Kommunikationsschnittstelle zwischen Smartphone und Sensoren. Die Kommunikation zum Smartphone erfolgt über die Serielle Schnittstelle, die Kommunikation zum Sensor über den I^2C-Bus.

CH340

Bei dem Arduino Chip CH340 handelt es sich nach dem Datenblatt [htt18] um einen BUS Konvertierungs Chip, der eine serielle USB Schnittstelle ansprechen kann. Es unterstützt sowohl 5V, als auch 3.3V Spannungsquellen zum Betrieb. Die Baudraten reichen von 50bps bis zu 2Mbps.

MPU-6050

Nach dem Datenblatt [Inc] besitzt der MPU-6050 Sensor ein Gyroskop und ein Beschleunigungssensor mit X-, Y- und Z-Achsen. Der Beschleunigungssensor kann dabei verwendet werden, um Erschütterungen zu erkennen.

Signalanalyse

Anschlagserkennung

Zur Erkennung von von Anschlägen und Abgrenzung von Hintergrundrauschen werden die Messdaten in Echtzeit geprüft. Hierzu werden die jeweils letzten beiden Messwerte miteinander verglichen. Ein Anschlag wird erkannt, wenn der aktuellere Wert den vorherigen Wert um ein bestimmtes Vielfaches der Standartabweichung übersteigt. Der Faktor sollte dabei auf das zu erwartende Rauschen abgestimmt werden - höhere Werte neigen weniger dazu, Ausschläge im Rauschen bzw. unabsichtliche Behrührungen der Oberfläsche als Anschlag zu erkennen, niedrigere Werte erlauben dagegen, auch sanfte Anschläge zu erkennen. In unseren Versuchen hat sich ein Faktor von 6 als stabil erwiesen.

Wurde ein Anschlag erkannt, so beginnt ein Zeitfenster von ca. 500ms, in dem eingehende Messwerte gepuffert werden. Nach Ablauf des Zeitfensters wird dem Programm signalisiert, dass ein Schlag aufgenommen wurde. Zusätzlich werden die Messwerte bis ca. 20ms vor dem erkannten Anschlag in die Zeitreihe aufgenommen.

Analyse der Beschleunigungsdaten

Die Klassifizierung aufgenommener Schläge erfolgt mittels Auswertung der stärksten Frequenzen. Hierzu werden die Messwerte zunächst einer diskreten Fouriertransformation unterzogen, die Spektrale Leistungs-

sdichte der Zeitreihe bestimmt. In der aktuellen Implementierung konnten Abtastraten von ca. 200Hz erreicht werden. Für ein Zeitfenster von ca. 500ms ergeben sich somit 128 Messwerte. Nach Anwendung der Fouriertransformation erhält man somit 64 diskrete Frequenzgruppen mit einer Trennschärfe von ca. 2Hz.

Aus diesen Daten werden die 4 Frequenzanteile mit der höchsten Leistungsdichte ermittelt zur weiteren Verwendung gespeichert.

Lernphase

Zum Erlernen einer Schlagposition werden 10 Schläge aufgezeichnet und der Durchschnitt der jeweils stärksten Frequenzgruppe, zweitstärksten Frequenzgruppe, usw... ermittelt. Diese werden als charakteristisches Pattern gespeichert.

Echtzeit-Klassifizierung

Ist die Echtzeit-Klassifizierung aktiviert, so werden nach erfolgter Anschlagserkennung und Schlagaufzeichnung die Messwerte analysiert und die stärksten Frequenzgruppen mit denen der gelernten Positionen verglichen. Der Schlag wird der Position mit der gerinsten totalen Abweichung zugeordnet. Erkennbar hierbei ist, dass in jedem Falle eine Zuordnung erfolgt - auch wenn die Abweichung absurd groß ist. Dies könnte jedoch leicht geändert werden, indem eine maximale Abweichung festgelegt wird und alle berechneten Abweichungen, die diesen Wert übersteigen, als nicht klassifizierbar eingestuft werden. Alternativ könnte, ähnlich zu anderen Clustering-Verfahren, eine Nachbarschaftsbeziehung zur Klassifikation verwendet werden.

Signalaufbereitung/Filterung

In der aktuellen Implementierung erfolgt keine Filterung oder Nachbereitung der Messwerte.

Eine Möglichkeit, um Rauschen zu unterdrücken wäre die Nutzung eines Grenzwertes. Auf diese Art können schwache Erschütterungen und leichte Vibrationen unterdrückt werden. Die Sensordaten könnten weiter geglättet werden durch Verwendung eines Box-Filters (Moving-Average) oder Gauss-Filters.

4. Diskussion

Literaturnachweis

[Her18] Felipe (felHR85) Herranz. Usbserial. https://github.com/felHR85/UsbSerial, 2018.

[htt18] http://wch.cn/. Usb to serial chip ch340. https://www.olimex.com/Products/Breadboarding/BB-CH340T/resources/CH340DS1.PDF, 2018.

[Inc] TDK Inc. Mpu 6000 and mpu 6050 product specification revision 3.4. https://www.invensense.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf.

[lu18] libpd (http://libpd.cc/). pd-for-android. https://github.com/libpd/pd-for-android, 2018.

[mwm18] mike w (mik3y). usb-serial-for-android. https://github.com/mik3y/usb-serial-for-android, 2018.

[uIoEMA18] IEM (http://iem.at/) Institute of Electronic Music and Acoustics. Pure data. https://puredata.info/, 2018.