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 |

1. Einleitung

Mit Hilfe von Piezosensoren, Kraftsensoren, soll es möglich sein von beliebigen Oberflächen Schwingungen aufzunehmen, die beim Klopfen auf das Objekt entstehen.

Das Klopfen soll abhängig von der Position auf dem Objekt und der Stärke einen unterschiedlichen DrumSound erzeugen. Zu den Möglichen Geräuschen zählen Trommeln, Basstrommel und Zimbal.

Die Trommelgeräusche werden mittels PureData synthetisiert. Die Geräusche werden in Abhängigkeit von der Entfernung und Stärke zum Piezosensor ausgewählt. Eine Mobile-App wertet die Daten des Piezosensors aus und sppielt das entsprechende Geräusch ab.

Fragestellungen

- Welche Materialien/Objekte eignen sich am Besten für AnyDrum?
- Wie genau lässt sich die Frequenz beim Tippen ermitteln?
- Wie gut lassen sich schwache Schläge von Starken unterscheiden?
- Wie gut ist die Lokalisationserkennung zwischen einem Piezo, zwei, drei und vier?

2. Konzept

3. 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* [3] verwendet.

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

4. Arduino

A CH340

[htt18]

B MPU-6050

5. Filterung

We use simple highpass filtering to sort out most of the noise and only let stronger vibrations from the surface pass the sensors...

To further smooth our data, we use boxfilter/ Gauss filter.

After basic filtering, we use ... to decide where the bang is coming from and what sound to play.

Instead of timbreID, which clusters and order features,we use ... [Bre]

DBSCAN is a density based cluster algorithm, which "cluster based on core and ...." In contrary to .... algorithm, such as K-Means, it has the advantage to detect arbritrary forms of cluster and filter out noise[EA13]

In fact, that we have much noise in our velocity data, we decided to use DBSCAN, which is easy to implement. To further increase accuracy and the unknown fact of epsilon, we further used ARCADE, which is a ... to .... [cite]

Using 3 sensors, we had .... results, while 2 ... .and 1 was unsatisfacory.

For 3 sensors we used barycentric interpolation, for 2 linear interpolation, for one we approximated by time/force?

Graph shows comparison between .... and the number of sensors used.

6. Diskussion

Literaturnachweis

[Bre] William Brent. A timbre analysis and classification toolkit for pure data. University of California, San Diego, Center for Research in Computing and the Arts.

[EA13] Mohammed T. H. Elbatta and Wesam M. Ashour. A dynamic method for discovering density varied clusters. In *International Journal of Signal Processing, Image Processing and Pattern Recognition Vol. 6, No. 1, February, 2013*, 2013.

[3] Felipe Herranz (felHR85). 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.

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