Vocal-Sampler

Projektdokumentation
Projekt-Typ: 2
Studienschwerpunkt: Musikinformatik

David Hanraths
Musik und Medien
907291

19/05/2022

Inhalt

Vorüberlegungen

Ausgehend von den Ergebnissen meiner vorangegangenen Projektarbeiten[1][2] wollte ich eine Möglichkeit finden, die darin erreichten Ergebnisse zu verklanglichen. Dafür sollte ein Instrument in SuperCollider programmiert werden, welches sich über eine GUI spielen bzw. steuern lässt.

Bisher wurden mithilfe des AudioTSNEViewers bereits folgende Funktionen erfolgreich umgesetzt:

- Ein Scatterplot wird angezeigt, dessen einzelne Punkte die Sprachsamples repräsentieren.
- Samples werden abgespielt, wenn der Mauszeiger sich über dem entsprechenden Punkt befindet.

Aus den anschließend gesammelten Metadaten meiner Library ergaben sich folgende weitere Potenziale:

- Die Punkte nach Emotion und Aussage zu sortieren.
- Ihre räumliche Nähe im Scatterplott akustisch darzustellen.
- Eine Form der Resynthese der Intonationsdaten umzusetzen.
- Eine grafische Repräsentation der Intonation der einzelnen Samples.

In diesem Projekt nahm ich mir vor, diese Potenziale zu erforschen, um einen Sampler für meine Sprachaufnahmen zu programmieren. Darin sollten besonders Möglichkeiten zur Interaktion mit der Library und dem tSNE-Scatterplot verwirklicht werden.

Mir war wichtig, dass der Patch leicht auf anderen Systemen umgesetzt werden kann. Da ich davon ausgehe, dass meine Sample-Library noch weiter wachsen wird, wollte ich außerdem darauf achten, dass der Code mit einer veränderten Library kompatibel bleiben würde.

SuperCollider

Anforderungen

Voraussetzung für die Reproduktion der Projektergebnisse ist, dass eine aktuelle Version von SuperCollider (3.11 oder später) unter GNU-Linux installiert ist. Ich nehme an, dass die Anwendung auch unter anderen unixartigen Betriebssystemen oder mit älteren SuperCollider-Versionen funktioniert, konnte beides jedoch bisher nicht testen.

Beide im Anhang befindlichen SuperCollider-Patches (s. *Anhang F & G*) sollten sich bei deren Ausführung im selben Verzeichnis befinden,
wie der Ordner “data” (s. *Anhang A*). Solange an der Struktur des Projektordners nichts verändert wurde, ist diese Bedingung bereits hergestellt.

[1] David Hanraths, *Verbal Interjection Sample Library - Projektdokumentation*, Robert Schumann Hochschule Düsseldorf - Institut für Musik und Medien, 12.05.2022.

[2] David Hanraths, *Audio Data-Mining t-SNE, Librosa & R-Intonation - Projektdokumentation*, Robert Schumann Hochschule Düsseldorf - Institut für Musik und Medien, 14.05.2022.

Der Patch **`tsne2dict.scd`** schafft die Voraussetzungen für den zweiten Patch **`vocal_sampler.scd`**, indem er folgende Funktionen erfüllt:

`[l. 01 – 50 ]:`

- Die Daten aus dem File "audiotsne.json" (s. *Anhang B*),
  welches die Zuordnung der Punkte und Pfade zu den Audiodateien beinhaltet, wird in ein **`Array`** aus **`Dictionaries`** übersetzt.
- Diese werden um die **`Keys`** "expression", "emotion", "csv" und "index", sowie deren entsprechende Werte (**`Values`**) ergänzt.
- Die bereits hinterlegten Pfade zu den Audiodateien werden an die lokale Ordnerstruktur angepasst.

`[l. 55 f.]:`

- Aus dem Array wird ein neues File mit dem Namen "vocalmap.json" geschrieben und im Unterordner "data" abgespeichert.

Solange der Pfad zum angehängten Ordner "data" unverändert bleibt, wird dieser Patch kein zweites Mal benötigt. Um ein Beispiel für das hierin erzeugte JSON-File geben zu können, habe ich der Anlage ein Beispiel beigefügt (s. *Anlage D*).

Sampler

Der zweite Patch **`vocal_sampler.scd`** besteht aus einem einzigen Code-Block, der einmalig ausgeführt werden muss. Darin wird zuerst das zuvor erzeugte File "vocalmap.json" in ein Array aus Dictionaries übersetzt. Deren Einträge beinhalten jeweils die Pfade zu einer Audidatei und einem CSV-File, die Koordinaten des Punktes auf dem Scatterplot, sowie eine Emotion und eine Aussage.
Die Begriffe "Eintrag", "Punkt" und "Sample" werden im Folgenden synonym füreinander benutzt.

Wird der Code-Block ausgeführt, sollte sich eine GUI öffnen, die den gesamten Bildschirm ausfüllt und auf der linken Seite verschiedene Steuerelemente darstellt:

1. **`[Button]`** *close / far / rand* - Jeweils nahegelegendster, entferntester oder ein zufälliger nächster Punkt - ausgehend von der Distanz zu den restlichen Punkten im Scatterplot.
2. **`[Slider]`** *Vocal ↔ Synth* - Das Mischverhältnis von Original (Audiosample) und Resynthese.
3. **`[Slider]`** *Speed:* - Die Frequenz, in der sich die Sequenz fortsetzt.
4. **`[Slider]`** *Starting Point:* - Neuer Ausgangspunkt beim Start der nächsten Sequenz.
5. **`[Button]`** *Play / Stop* - Startet / Stoppt die Sequenz.

Bevor der Sampler über den "Play"-Button gestartet werden kann, muss per Mausklick eine Auswahl an Emotionen und Aussagen getroffen werden. Daraufhin beginnt der Sampler eine Sequenz abzuspielen, die am "Starting Point" beginnt und sich fortsetzt, bis keine Einträge mehr zur Verfügung stehen. In diesem Fall stoppt der Sampler und setzt den Slider "Starting Point" auf einen zufälligen Punkt zurück. Die durch die abgespielte Sequenz entstandene Punktekarte wird dargestellt, bis der Sampler durch Klicken von "Play" neu gestartet wird.

Der nächste Punkt in der Sequenz wird anhand der Einstellungen des Buttons “close/far/rand” und der getroffenen Auswahl an Emotionen und Aussagen ausgesucht und das entsprechende Soundfile und die Resynthese werden abgespielt. Gleichzeitig wird in einer zufälligen, aber distinktiven Farbe und entsprechend seiner Koordinaten der zugehörige Punkt im Scatterplot dargestellt.

Außer “Starting Point” können alle Parameter genutzt werden, um das Verhalten Samplers zu beeinflussen, während er läuft.

Fazit

Dieser Code hat mich vor verschiedene Herausforderungen gestellt. Die Grundfunktionen des Samplers waren schnell hergestellt, doch besonders die einzelnen Details umzusetzen und die Stabilität herzustellen erforderte viel “Debugging”. Darüber habe ich viel Neues über SuperCollider’s GUI-Funktionen und über Routinen gelernt.

Die Punkte im Scatterplot sind nun - anders als im “AudioTNSEViewer” - farblich unterschiedlich markiert und somit erstmals visuell differenzierbar. Außerdem ergeben sich aus der hier erreichten Form der Vertonung unzählige interessante musikalische Muster und Formen. Daraus eröffnen sich mir neue Perspektiven auf die von mir gesammelten Samples.

Derzeit sehe ich für meine weiterführende Arbeit an diesem Projekt noch folgende unerforschte Potenziale:

1. Nach wie vor die grafische Repräsentation der Intonationsdaten (s. **Vorüberlegungen**)
2. Weitere Formanten (F1, F2, etc.) zur Resynthese zu nutzen.
3. Die Ausdrücke `["yes", "no", "question"]` auf der Karte ebenfalls visuell unterscheidbar zu gestalten.
4. Die Abstände zwischen den Punkten zusätzlich darzustellen, zum Beispiel in Form unterschiedlich starker Verbindungslinien zwischen den Punkten.
5. “Pause”- und “Loop”-Funktionen, sowie die “MouseOver” Funktion des AudioTSNEViewer zu integrieren.
6. Eine 3D-Version dieses Samplers umzusetzen.
7. Eine `MIDIOut`-Funktion zu implementieren, die es ermöglicht, externe Hardware mit den F0-Daten zu adressieren.
8. Mehr Möglichkeiten zur Rhythmisierung bereitzustellen.

Zeitlicher Aufwand

Die Umsetzung dieser Projektarbeit hat insgesamt etwa 67 Arbeitsstunden in Anspruch genommen, die sich wie folgt zusammensetzen:

- Konzeption ~8 Std.
- Coding & Debugging ~51 Std.
- Dokumentation ~8 Std.

Anhang

```
Vocal-Sampler
├── data -------------------- (A)
│   ├── audiotsne.json ------ (B)
│   ├── csv ----------------- (C)
│   ├── vocalmap.json.xmpl -- (D)
│   └── Wave ---------------- (E)
├── tsne2dict.scd ----------- (F)
└── vocal_sampler.scd ------- (G)
```

Quellen

Audio Data-Mining
David Hanraths, *Audio Data-Mining t-SNE, Librosa & R-Intonation - Projektdokumentation*, Robert Schumann Hochschule Düsseldorf - Institut für Musik und Medien, 14.05.2022.

Verbal Interjection Sample Library - Projektdokumentation
David Hanraths, *Verbal Interjection Sample Library - Projektdokumentation*, Robert Schumann Hochschule Düsseldorf - Institut für Musik und Medien, 12.05.2022.