Johannes Kepler Universität Linz
Institut für Computational Perception

Praktikum aus Informatik

LVA #344.025

Whitepaper

Disc-JoQey

Ein (halb-) automatischer Ansatz zur Extraktion von Musikdateien aus Schallplattenaufnahmen

Autor: Christian Franz Junker, Bakk.
M.Nr. 08596949, SKZ 921

Betreuung: Dipl.-Ing. Peter Knees

Sommersemester 2009

Inhaltsverzeichnis

1. Einleitung

> *You listen to these modern records, they're atrocious, they have sound all over them. There's no definition of nothing, no vocal, no nothing, just like - static.*
>
> Bob Dylan

Schallplatten sind ein analoges Speichermedium für Töne und Musik. Sie bestehen aus einer flachen Scheibe, auf der eine spiralförmige Rille (meist) von außen nach innen verläuft und die gespeicherte Information auf die Nadel eines Tonabnehmers überträgt [1].

Schallplatten waren im zwanzigsten Jahrhundert der Haupttonträger für kommerzielle Musik. Allerdings wurden sie ab Ende 1980 weitestgehend von digitalen Medien und ihren, an sich klar überlegenen Klangeigenschaften, aus dem Mainstream verdrängt. Sie wurden aber weiterhin produziert und werden heutzutage, hauptsächlich von DJs und Hobbyisten, immer noch gerne eingesetzt und erfreuen sich in letzter Zeit generell wieder steigender Beliebtheit. [2]

Warum müssen Schallplatten archiviert werden?

- Verschleiß: Schallplatten zerkratzen aufgrund des weichen Materials sehr leicht und sind anfällig gegenüber Staub, da sie sich statisch aufladen. Ferner wird durch das Abspielen selbst die Modulation in der Rille verändert. Auch können sie sich durch Hitze verformen.
- Unhandlichkeit: Dieser Punkt ist sicherlich Ansichtssache, Aussehen und Haptik von Schallplatten tragen sicher einen großen Teil zu ihrem Reiz bei. Klar ist allerdings, dass durch das Archivieren ganze Sammlungen auf einem kleinen MP3-Player leicht transportiert werden können.
- Segmentierung: Zusätzlicher Komfort kann durch Segmentierung der Aufnahme gewonnen werden, da dies erlaubt Stücke beliebig zu mischen und abzuspielen.

Audioaufnahmen von Schallplatten weisen allerdings einige, eine Archivierung erschwerende, Eigenheiten auf, die so bei digitalen Medien, wie CDs, nicht auftreten.

So handelt es sich hier normalerweise um einen kontinuierlichen Datenstrom. Eine Schallplatte trägt ferner auch meist auf beiden Seiten Daten, was die Aufnahme in zwei Teile trennt. Auf einer CD hingegen sind die Stücke normalerweise bereits als einzelne Dateien gespeichert.

Eine weitere prominente Eigenschaft von Schallplattenaufnahmen sind Nebengeräusche wie Rauschen, Knacken, etc., die sich aus beschriebenem Verschleiß und Staub ergeben. Diese Geräusche erschweren die Analyse der Aufnahme, da z.B. eigentlich stille Passagen evt. nicht erkannt werden können.

Ein generelles Problem einer kontinuierlichen Aufnahme (ohne a priori Wissen) ist, dass kaum unterschieden werden kann ob eine stille Passage zum Stück gehört, oder die Pause zwischen zwei Stücken kennzeichnet.

Auch kann es vorkommen, dass Stücke nahtlos ineinander übergehen und es somit fast ausgeschlossen ist diese Stücke zu segmentieren. Ein reales Beispiel, in der diese Situation auftritt, wäre das Album “The Wall” von Pink Floyd, in dem Soundeffekte zwischen den Tracks überleiten.

Wie lässt sich nun also eine Archivierung realisieren? Eine mächtige, aber auch aufwendige Möglichkeit, ist die Verwendung eines Audioeditors wie z.B. Audacity. Abbildung 1 zeigt einen Screenshot dieser Anwendung.

Solche Audioeditoren erlauben es Aufnahmen mit einer Vielzahl von Werkzeugen zu bearbeiten, zu segmentieren, in verschiedene Formate zu exportieren, etc. Allerdings muss alles vom Benutzer selbst spezifiziert werden, was einen komplexen und zeitaufwendigen Workflow zur Folge hat.

Hier setzt diese Arbeit an. Es wird eine Applikation entwickelt, die versucht es dem Benutzer so einfach wie möglich zu machen seine Schallplatten zu archivieren, ohne zu viel an Flexibilität bei der Erstellung, im Vergleich zur Verwendung eines Audioeditors, einzubüßen.

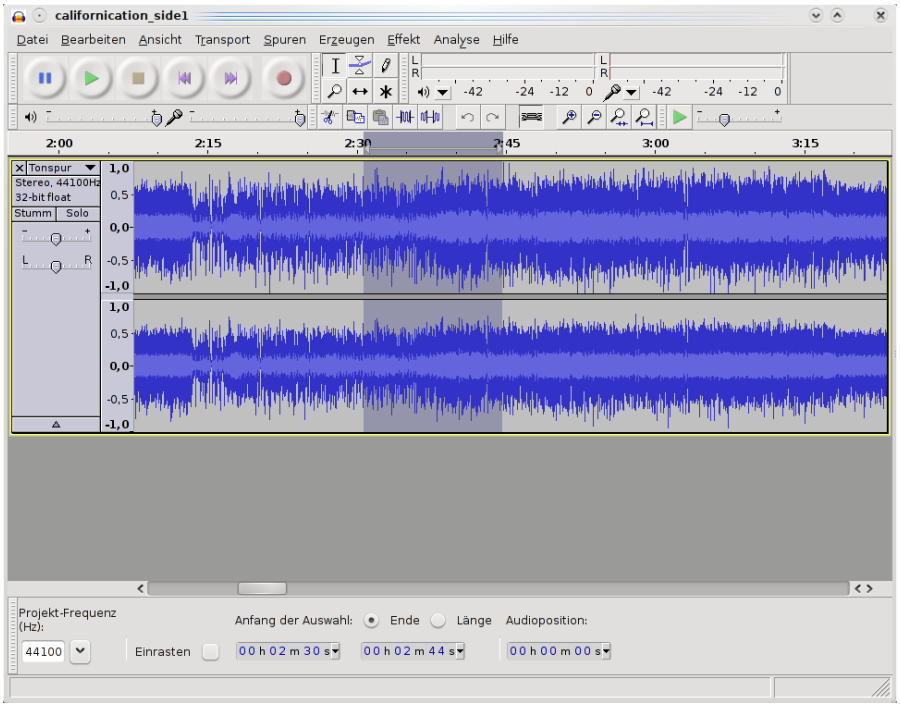


Abbildung 1: Screenshot des Audioeditors Audacity.

2. Problemstellung und Zielsetzung

2.1. Problem

Wie bereits in der Einleitung dargestellt, ist das Segmentieren einer Schallplattenaufnahme ist eine nichttriviale Aufgabe. Anders als bei digitalen Aufnahmen, in denen zwischen zwei Stücken absolute Stille herrscht, bzw. die Stücke schon segmentiert vorliegen, kann bei analogen Aufnahmen oft nicht zwischen Stille und einem Teil des Stücks unterschieden werden.

Ferner ist ohne Kenntnis des Stückes kaum entscheidbar, ob es sich bei stillen Passagen um eine Pause im Stück, oder um die Trennung zwischen zwei Stücken handelt.

2.2. Ziele

Es soll eine Applikation entwickelt werden, die den Anwender dabei unterstützt die Tracks aus Aufnahmen von Schallplatten zu segmentieren und in MP3 Dateien zu extrahieren. Ferner sollen diesen MP3 Dateien Metadaten hinzugefügt werden können.

Um dies zu erreichen sollen Metadatensuchmaschinen verwendet werden. Diese liefern die Informationen zu den Aufnahmen, die der Anwender anschließend, in einem halbautomatischen Ansatz, kontrolliert und gegebenenfalls korrigiert.

2.3. Open Source: BSD Lizenz

Aufgrund der edukativen Natur dieser Arbeit, wird die Applikation als quelloffenes Projekt (Open Source) entwickelt. Als Lizenz, die die Nutzungsbestimmungen definiert, wird die BSD Lizenz, eine der einfachsten und liberalsten [3] - und somit für diesen Zweck wohl am besten geeignete Lizenz, gewählt.

In Anhang A ist die Lizenz abgedruckt, so, wie sie für das gesamte Projekt, inklusive dieses Whitepapers, gültig ist.

Die Bestimmungen lassen sich wie folgt zusammenfassen [3]:

- Das Copyright muss intakt bleiben. Es darf nicht behauptet werden, dass das Projekt von jemand anderem als dem Copyright Holder erarbeitet wurde.
- Der Name des Autors oder des Projekts darf nur nach vorheriger, schriftlicher Erlaubnis des Autors verwendet werden, um Derivate[1] zu bewerben.
- Der Entwickler kann unter keinen Umständen für Schäden belangt werden, die sich, in welcher Form auch immer, durch das benutzen dieses Projekts ergeben.

Sofern diese Bestimmungen eingehalten werden, kann das Projekt, in jeglicher Form, für eigene Zwecke verwendet werden.

[1] Entwicklungen basierend auf diesem Projekt.

3. Stand der Technik

Im Windows - Bereich gibt es einige Programme die sich der Schallplattenarchivierung verschrieben haben. Dies Programme sind allerdings zumeist proprietär und unterstützen eher selten die Verwendung von Metadaten aus dem Internet für die Segmentierung, bzw. können diese teilweise überhaupt nicht abfragen.

Eine Auswahl an verfügbaren Programmen für Windows kann auf folgender Webseite gefunden werden: http://www.filebuzz.com/files/Convert_Vinyl_MP3/1.html

Im Linux/Unix-bereich, bzw. generell im quelloffenen Bereich, gibt es allerdings kaum Programme die speziell für dieses Anwendungsgebiet entwickelt wurden. Hier muss zwangsläufig auf komplexere Audioeditoren, wie z.B. Audacity, zurückgegriffen werden.

3.1. GramoFile

GramoFile [4] ist eine Applikation primär für Linux/UNIX, die es erlaubt Audioaufnahmen zu erstellen, diese zu bearbeiten und zu segmentieren.

Die Applikation ist Terminal-basiert, weist aber bereits Züge einer graphischen Benutzeroberfläche auf.

Die Applikation unterstützt Recording, bietet eine Abspielmöglichkeit und kann mit sehr großen Dateien umgehen.

Unterstützt wird ferner bereits eine Bearbeitung der Audiodatei durch Filter. Neun solcher Filter werden mitgeliefert. Diese Filter können beliebig kombiniert verwendet werden. Außerdem ist eine Parametrisierung dieser Filter möglich.

Die Segmentierung einer Aufnahme in einzelne Tracks wird ebenfalls unterstützt. Die Prozessierung erfolgt dabei on the fly ohne die Verwendung temporärer Dateien.

Leider ist das Programm allerdings mittlerweile nicht mehr betreut. GramoFile wurde zuletzt im Mai 2001 aktualisiert.

3.2. VinylStudio

VinylStudio [5] ist ebenfalls ein Programm um Schallplatten zu digitalisieren. Das Programm ist allerdings Windows exklusiv und kostenpflichtig.

Das Programm erlaubt ebenfalls Aufnahmen von analogen Tonträgern anzulegen, diese zu bearbeiten und in Tracks zu exportieren.

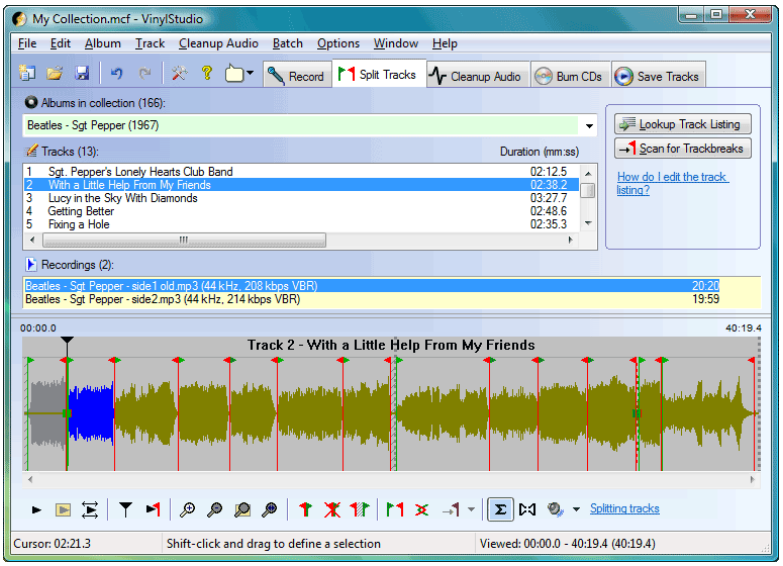


Abbildung 2: Screenshot der Anwendung VinylStudio

Im Gegensatz zu GramoFile bietet VinylStudio allerdings eine anwenderfreundliche graphische Benutzeroberfläche. In Abbildung 2 ist ein Screenshot[2] der Anwendung dargestellt.

Hervorzuheben ist, dass die Anwendung Metadaten im Internet abfragen kann und diese auch verwendet, um die Tracks grob vor zu segmentieren.

[2] http://www.alpinesoft.co.uk/VinylStudio/edit_screen.aspx

4. Werkzeuge

Um die geplante Architektur umzusetzen wurden die folgenden Werkzeuge verwendet. Wie auch bei der Applikation selbst, handelt es sich hierbei um Open Source Software, bzw. Software mit kombinierten Modelle (Qt).

4.1. Plattform: C++/Qt

Die Applikation wird in C++ entwickelt und setzt als Haupttoolkit Qt Version 4.5 [6] ein. Qt ist längst kein reines Toolkit mehr sondern unterstützt neben den GUI Komponenten zahlreiche Funktionen wie Datenbankzugriff, Mutlithreading, OpenGL, etc. um nur eine Auswahl zu nennen.

Qt wird unter einem kombinierten Lizenzmodell angeboten. Neben einer kostenpflichtigen proprietären Lizenz (schließt u.a. Support mit ein), auch unter freien Lizenzen (LGPLv2.1 und GPLv3.0).

Die Philosophie hinter Qt ist "write once, compile anywhere" erreicht wird dies unter anderem dadurch, dass Wrapper für nahezu alle benötigten Datentypen und Funktionen existieren und so Plattformunabhängigkeit garantiert wird.

Unterstützt wird Qt unter anderem auf folgenden Plattformen:

- Apple Mac OS X (32 Bit)
- Linux (32 and 64 Bit)
- Microsoft Windows

Eine vollständige Aufstellung aller unterstützten Plattform ist in der Dokumentation unter Supported Platforms[3] zu finden.

[3] http://doc.trolltech.com/4.5/supported-platforms.html

4.2. Audiodateiverarbeitung: libsndfile

Libsndfile [7] ist eine C Library um Dateien, die (gesampleten) Sound beinhalten, wie z.B. Microsoft WAV, zu bearbeiten. Veröffentlicht ist libsndfile unter der GNU Lesser General Public License.

Die Library wurde zwar primär für Linux Systeme entwickelt, funktioniert allerdings auch auf anderen Systemen, wie z.B. Windows und Mac OS.

Es wird eine Vielzahl von Formaten z.B. Microsoft WAV, SGI/Apple AIFF/AIFC, Headerless RAW, etc. in den unterschiedlichsten Encodings wie z.B. unsigned 8-32 Bit, 32/64 Bit Float, etc. unterstützt. Die volle Tabelle kann auf der Projekthomepage[4] gefunden werden.

Die Library erlaubt ferner eine on the fly Konversion zwischen den unterstützten Dateiformaten. Dies geschieht transparent, der Benutzer kann also einfach mit den Dateien arbeiten, ohne auf das Format Rücksicht nehmen zu müssen.

4.3. Metadaten Service: musicbrainz3

MusicBrainz [8] ist ein Metadatendienst im Internet. Er stellt einen Katalog mit Informationen zu Musik Veröffentlichungen, wie Namen des Künstlers, die Trackliste, etc., zur Verfügung. Verwaltet und erweitert wird MusicBrainz von den Benutzern.

Neben der Webseite, bzw. dem Webservice und eigenständigen Programmen, wird auch eine Library [9] angeboten um auf den Service zuzugreifen. Diese Library liegt in zwei Versionen vor:

- Version 2.x: basiert auf dem RDF-Webservice. Diese Version ist noch am weitesten verbreitet, ist allerdings als deprecated markiert.
- Version 3.x: basiert auf dem neuen XML-Webservice[5]. Es wird von den Entwicklern empfohlen für Neuentwicklungen diese Version zu verwenden.

[4] http://www.mega-nerd.com/libsndfile/
[5] Details zur Spezifikation unter http://wiki.musicbrainz.org/XML_Web_Service.

Die Bibliothek steht für Linux, Mac Os und Windows (offiziell bis Windows XP, 32 Bit) zur Verfügung. Die API ist unter http://users.musicbrainz.org/~luks/docs/libmusicbrainz3/ zu finden.

4.4. MP3 Export: Lame

Lame [10] ist ein quelloffener (LGPL) MP3 Encoder von hoher Qualität. Lame ist zwar an sich selbst ein Tool, bietet aber eine Library an, um die Funktionen in anderen Programmen zu verwenden.

Unterstützt werden vom Tool MPEG 1, 2 und 2.5 layer III (MP3) En- bzw Dekodierung. Für die Kodierung kann entweder das CBR (konstante Bitrate) Verfahren oder eines der zwei angebotenen variablen Verfahren (VBR, ABR) verwendet werden.

Um die Usability zu steigern bietet Lame ferner eine Reihe von Presets an, die für verschiedene Zwecke optimiert sind und keine weiteren Einstellung benötigen

Lame ist ein sehr effizienter Encoder. So behauptet ein Beispiel des Herstellers, dass auf einem Pentium II Prozessor mit 266Mhz auf höchster Qualitätsstufe schneller als in Echtzeit encodiert werden kann.

Auch die Qualität ist, laut den Autoren, in vielen Fällen besser als die meisten konkurierenden Tools, sowohl quelloffen als auch proprietär.

4.5. ID3 Tags: id3lib

Id3lib [11] ist eine quelloffene Bibliothek für das Lesen, Schreiben und Bearbeiten von ID3 Tags. Sowohl ID3v1, als auch ID3v2 werden unterstützt.

Hauptziele der Bibliothek sind die Konformität zum ID3v2 Standard, Portabilität, sowie eine einfache und trotzdem mächtige API.

Die Bibliothek erlaubt die Arbeit mit ID3 Tags auf einem hohen Abstraktionslevel und unterstützt von Haus aus häufig gebrauchte Arbeitsschritte wie z.B. Formatkonversion.

Id3Lib unterstützt als Sprachen hauptsächlich C, sowie C++. Zusätzlich existiert ein COM[6] Wrapper, der es allen COM unterstützenden Sprachen erlaubt die Bibliothek zu nutzen.

[6]Component Object Model. Plattform Technologie zur dynamischen Objekterzeugung und Interprozesskommunikation unter Windows.

5. Implementierung

Im Folgenden werden die wichtigsten Implementierungsdetails beschrieben. Der Fokus dieser Arbeit erlaubt allerdings nicht alle Einzelheiten zu beleuchten, weshalb allgemein auf die verwendeten Konzepte eingegangen wird. Für einen tieferen Einblick in die Implementierung muss daher auf den Sourcecode verwiesen werden.

5.1. Handhabung der Datenmenge

Audioaufnahmen von Schallplatten sind meist große Dateien, zum Beispiel braucht eine normale Standardaufnahme der ersten Seite der Schallplatte "Californication" der Band Red Hot Chili Peppers[7] ca. 166 Megabyte an Speicherplatz.

In der Applikation werden zwei solche Dateien verwendet, also in dieser Größenordnung dann insgesamt ca. 330-340 Megabyte. Um diese Datenmenge nun nicht ständig im Hauptspeicher halten zu müssen, werden zwei Konzepte eingesetzt: eine *Pipeline* soll erst die Daten laden wenn diese auch wirklich gebraucht werden und auf die Daten selbst wird über einen *AudioBuffer* zugegriffen, der dynamisch Daten aus der Datei nachlädt.

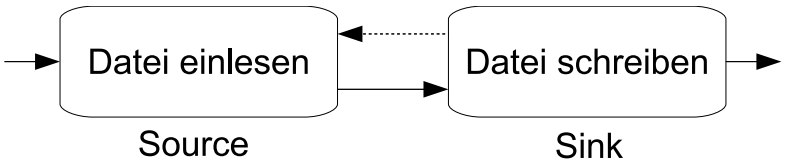
5.1.1. Pipeline

Die Applikation basiert auf einem Pipeline Ansatz, der von einem ähnlichen Ansatz inspiriert wurde, der im ITK[8] zum Einsatz kommt [12].

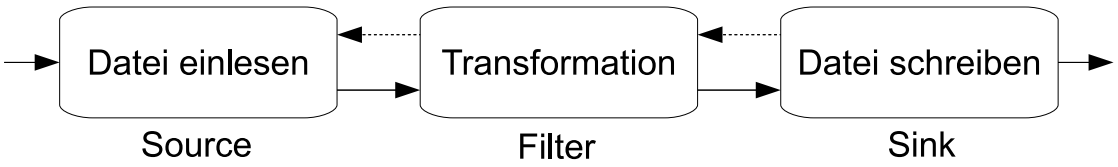
Der Datenfluss der Pipeline startet immer in einem *Source* Element, das die Daten bereitstellt und endet in einem *Sink* Element, das die Daten anfordert. Die kürzeste Pipeline besteht somit aus zwei Elementen, z.B. Datei einlesen → Datei in anderem Format abspeichern (dargestellt in Abbildung 3(a)).

Zusätzlich zu *Source* und *Sink*, existiert das *Filter* Element. Dieses kann sowohl als *Source*, als auch als *Sink* in der Pipeline agieren. Dies erlaubt nun Pipelines die aus

[7] Diese Aufnahme wurde vom Autor erstellt und kam zum Testen der Applikation zum Einsatz.
[8] Insight Segmentation and Registration Toolkit (www.itk.org).



(a) Pipeline: Source → Sink



(b) Pipeline: Source → Filter → Sink

Abbildung 3: Pipelining im Überblick. Datenfluss - durchgehende Linie, Kontrollfluss - gestrichelte Linie.

mehr als zwei Elementen bestehen, z.B. Datei einlesen → Datei transformieren → Datei abspeichern (dargestellt in Abbildung 3(b)).

Die Bausteine der Pipeline sind generisch implementiert, was es erlaubt, das Konzept auf beliebige Datentypen anzuwenden. Ein *Source* spezifiziert den Ausgabetyp, ein *Sink* den Eingabetyp und ein *Filter* beides. Kombinierbar sind nur Elemente, die Übereinstimmende Eingabe/Ausgabe -, bzw. Ausgabe/Eingabe Typen aufweisen.

Der Vorteil bei dieser Architektur ist, dass die Daten erst prozessiert werden, wenn diese auch wirklich gebraucht werden.

Der Kontrollfluss ist nämlich vom *Sink* zur *Source* (bzw. *Filter*). Wenn also, um beim obigen Beispiel zu bleiben, am Sink das Speichern der Daten veranlasst wird, werden die Daten vom korrespondierenden *Source* bzw. *Filter* angefordert. Ein *Filter* fordert dann wieder die Daten von seiner *Source* bzw. *Filter* an und so weiter.

5.1.2. AudioBuffer

Der *AudioBuffer* wird verwendet, um auf die Daten der Audioaufnahmen zuzugreifen. Die Klasse ist generisch, wodurch alle Formate, die von der libsndfile Bibliothek bearbeitet werden können, unterstützt werden.

Um die großen Datenmengen bearbeiten zu können, wird ein Cache-Ansatz [13] verfolgt, der einen Mittelweg zwischen Datenhaltung komplett im Hauptspeicher und Lesen von der Festplatte darstellt.

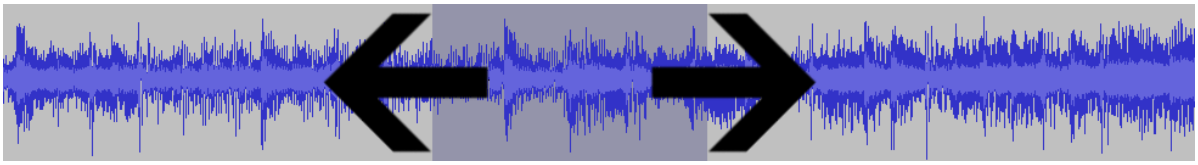


Abbildung 4: Idee Cache-Ansatz

Abbildung 4 zeigt die Idee dieses Ansatz. Es wird ein (beliebig großes) Fenster über die Daten gelegt. Dieses Fenster repräsentiert einen gewissen Datenausschnitt aus der Datei, der durch die Größe des Fensters begrenzt wird. Dieser Ausschnitt wird in den Hauptspeicher geladen und so lange gehalten, bis auf Daten außerhalb des Fensters und somit Daten, die sich nicht im Hauptspeicher befinden, zugegriffen wird. In diesem Fall wird das Fenster entsprechend nach rechts/links "verschoben" (bzw. wird der Ausschnitt geladen, in dem sich die benötigten Daten befinden).

Dieser Ansatz kombiniert die Vorteile beider Ansätze (Festplatte, Hauptspeicher):

- Höhere Geschwindigkeit als beim Lesen nur von der Festplatte durch Reduzierung der I/O Zugriffe.
- Geringere Hauptspeicherauslastung, da nur einzelne Ausschnitte und nicht die gesamten Daten gehalten werden.

Der Ansatz bleibt allerdings, wie erwähnt, ein Mittelweg und kann aufgrund der Struktur nie das volle Potential der einzelnen Ansätze ausschöpfen.

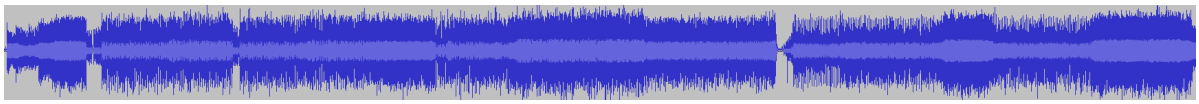
5.2. Waveform

Um die Waveform zu implementieren wird ein Ansatz verwendet der in einer ähnlichen Form in vielen Audioeditoren, z.B. Audacity (nachzulesen im Source Code) - Abbildung 5(a), zum Einsatz kommt.

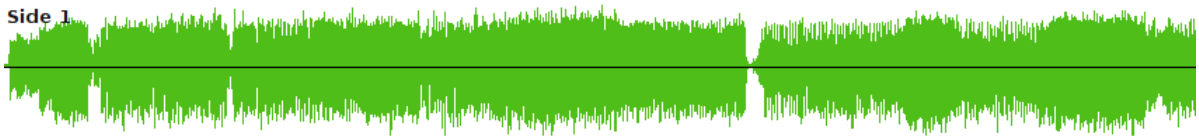
So genannte "Peak" oder "Overview" Files führen eine starke Datenreduktion auf dem Audiostream durch und liefern trotzdem sehr gute Ergebnisse.

Die Grundidee ist folgende: In einem Fenster über dem Datenstrom wird der maximale - und minimale Peak[9] bestimmt. Das Fenster wird nun durch diese zwei Eigenschaften repräsentiert.

In der graphischen Waveform werden nun auf einer X-Achse nacheinander (hier 1 Pixel breit) die Fenster aufgetragen, repräsentiert durch eine senkrechte Linie zwischen dem maximalen - und minimalen Peak.



(a) Waveform - Audacity



(b) Waveform - Disc-JoQey

Abbildung 5: Waveforms im Vergleich

Interessante Punkte dieser Implementierung:

- Die "Peaks" werden on the fly (in verschiedenen Auflösungsstufen) generiert und nicht separat in eine Datei abgespeichert.
- Die maximale erreichbare Auflösung liegt bei zwei Samples, da ausschließlich auf das "Peak" Konzept gesetzt wird. Eine höhere Auflösung ist für den Fokus dieser Applikation nicht angebracht.

In Abbildung 5 wird die Implementierung im Vergleich zum Audioeditor Audacity dargestellt.

5.3. AudioPlayer

Der implementierte *AudioPlayer* basiert auf dem Multimediaframework Phonon und ist als *Sink* implementiert.

[9] Positiver bzw. negativer Ausschlag

5.3.1. Phonon

Phonon ist ein Multimediaframework, das für die Desktop Umgebung KDE 4 entwickelt wurde. Dieses Framework wurde von Trolltech in Qt integriert und erweitert, um es auf allen wichtigen Plattformen lauffähig zu machen [14].

Phonon bietet durch eine Aufsplittung in Schnittstelle und Backends einen zentralen Zugangspunkt, der von der direkten Verwendung der Soundmodalität (z.B. Alsa unter Linux) abstrahiert. Dadurch kann auch eine zentrale Konfigurationsmöglichkeit für Audio/Video geboten werden.

Ferner wird automatisch vom Framework die beste Modalität für die Medienwiedergabe gewählt, was es vereinfacht in Umgebungen zu arbeiten, in denen z.B. mehrere Soundkarten (USB Headsets, USB Boxen, etc.) installiert sind.

Abbildung 6: Konzept von Phonon

Abbildung 6 zeigt das Konzept [15] des Phonon Frameworks. Hauptkomponenten der Architektur sind ein *MediaObject*, das den Sound "produziert", sowie ein *AudioOutput* (bzw. *VideoOutput*) der den Sound über die Peripherie des Computers ausgibt. Verbunden sind diese Elemente durch einen *Path*, in den zusätzlich Effekte eingefügt werden können.

Verschiedene Backends, wie z.B. Xine, DirectShow, QuickTime etc., werden durch Phonon unterstützt. Eine Liste[10] kann auf der Projekthomepage gefunden werden.

5.3.2. Wrapper

Um nun die Konzepte der Phonon Technologie in die Applikation zu integrieren und den *AudioPlayer* in eine Pipeline integrieren zu können, wurde ein Wrapper implementiert.

[10] http://phonon.kde.org/cms/1034

Dieser Wrapper bietet eine sehr ähnliche Schnittstelle wie das Phonon Gegenstück, vereinigt aber den kompletten Workflow um einen Audiooutput zu erstellen in einem *Sink*, der in eine Pipeline eingebaut werden kann. Es muss nun nur die Hauptsource der Applikation, die die Repräsentation des Audiofiles liefert, mit dem *AudioPlayer* verbunden und dieser aktualisiert werden um die Datei abspielen zu können.

5.4. MetaData

5.4.1. Verwaltung in der Applikation

Die Metadaten werden für das bearbeitete Projekt in einer Datenstruktur *MetaData* gespeichert.

Diese Datenstruktur speichert folgende globale Metadaten:

- Album
- Künstler
- Genre
- Jahr
- Kommentar

Die Informationen zu den einzelnen Tracks befinden sich in einer Liste (*TrackList*). Um auf diese *TrackList* zuzugreifen, wird das *MetaData* Objekt verwendet.

Die Tracks in der *TrackList* müssen keiner besonderen Reihenfolge unterliegen, so können Tracks von den zwei verschiedenen Seiten gemischt vorliegen, allerdings bestimmt die Position in dieser Liste, abhängig von der Seite, die Track Nummer. Beim Bearbeiten über die GUI wird sichergestellt, dass die Reihenfolge mit der Angezeigten übereinstimmt.

Zu jedem Track werden folgende Informationen gespeichert:

- Seite
- Titel
- Start (in Millisekunden)

- Stop (in Millisekunden)

5.4.2. Metadatenabfrage bei Musicbrainz

Verwendet wird, wie bereits erwähnt, die XML-Webservice-basierte Version 3 der MusicBrainz Library.

Die Schnittstelle um Metadaten bei Internetdiensten abzufragen ist generisch gehalten und erlaubt prinzipiell nach allen globalen Metadaten die in 5.4 dargestellt wurden zu suchen. Aufgrund von Einschränkungen kommen für die Anfrage bei MusicBrainz nur Albumtitel und Künstlername zum Einsatz.

Bei der Anwendung der Library wurde allerdings ein Problem bezüglich Sonderzeichen in der Verarbeitung festgestellt.

Die Library arbeitet intern mit dem Datentyp *std::string* der Standard C++ Library. Laut Standard basiert *std::string* auf ASCII und ist byte- und nicht zeichenorientiert. Der Webservice erwartet allerdings anscheinend UTF8 Zeichenfolgen.

Die Auswirkungen dieses Sachverhalts sind wie folgt: Werden UTF8 Sonderzeichen in einem *std::string* abgespeichert, werden für z.B. ein *ö* statt 1 Byte (ASCII) 2 belegt (UTF8). Dies verändert das Verhalten der Kodierung der Sonderzeichen. Das *ö* wird z.B. statt dem eigentlich korrekten ASCII-Wert %f6, in den Wert %C3%B6 umgewandelt, da *std::string* wie erwähnt ASCII orientiert ist und so die 2 Byte des UTF8 getrennt betrachtet.

Zur Veranschaulichung folgen Beispiele mit dem Sonderzeichen *ö*. Es wird nach dem Album "Volta" der Künstlerin Björk direkt am Webservice[11] gesucht:

- Korrekte ASCII Kodierung:
  ?type=xml&artist=Bj%f6rk&title=Volta
  → `403 BAD REQUEST`
- Erwartete Kodierung:
  ?type=xml&artist=Bj%C3%B6rk&title=Volta
  → normales Ergebnis

[11] http://musicbrainz.org/ws/1/release/

Der Versuch dieses Verhalten händisch nachzuahmen scheiterte allerdings, da es nicht möglich war das %-Zeichen zu escapen, was Resultate wie %25C3%25B6 (für das *ö*) zur Folge hatte. Es musste daher auf entsprechende (Um)Codierungen im Programm zurückgegriffen werden, die allerdings nur in Systemen mit eingestelltem UTF8 Zeichensatz funktionieren (getestet unter: Arch Linux - 64 Bit, Kubuntu installiert - 32 Bit, Kubuntu Live CD - 32 Bit; jeweils UTF8).

5.5. MP3Writer

Der *MP3Writer* ist dafür zuständig, basierend auf der Tracklist, die Audiodatei in einzelne Tracks zu exportieren. Er ist als *Sink* implementiert und steht so am Ende der Pipeline.

Für die zwei Seiten der Schallplatte, die als eigene Files vorliegen, existiert je eine eigenständige Pipeline die in der jeweiligen *Source* starten und über die globale Metadatenbank gesteuert werden.

In der Metadatenbank sind, wie beschrieben, die globalen Metadaten, sowei die Trackliste gespeichert. Die Trackliste wird abgearbeitet und die zur Seite gehörenden Stücke aus dem Audiofile gelöst und in MP3 Dateien exportiert. Hierzu kommt die Lame Library zum Einsatz.

Als Verfahren für die Bitrate kann entweder VBR (variable Bitrate), oder CBR (konstante Bitrate) verwendet werden. Im VBR Verfahren wird, das Stück analysiert und, je nach Komplexität, einzelne Zeitabschnitte mit unterschiedlich hoher Bitrate codiert, während bei CBR die eingestellte Bitrate für das gesamte Stück verwendet wird [16]. VBR ermöglicht daher, trotz gleicher Qualität, die Dateigröße zu reduzieren.

Um die VBR-Bitrate einzustellen, können drei Parameter angegeben werden: minimale Bitrate, mittlere Bitrate und maximale Bitrate. Die an den *MP3Writer* übergebene Bitrate wird als mittlere Bitrate eingestellt. Maximum und Minimum werden über folgende Annahme begrenzt: $Max = 2 * Bitrate$ bzw. $Min = 0.5 * Bitrate$.

Um die so erstellten MP3 Dateien mit ID3 Tags zu versehen kommt id3lib zum Einsatz. Diese Library bietet eine bessere Schnittstelle und höhere Funktionalität, als der in Lame integrierte Support.

6. Ergebnisse

Im Folgenden werden die Applikation - speziell das GUI - und ihre Komponenten aus Benutzersicht vorgestellt.

6.1. Applikation im Überblick

Die entwickelte Applikation besteht im Wesentlichen aus zwei Hauptkomponenten, die in Abbildung 7 dargestellt sind - einer Trackverwaltung (a), sowie einer Exportseite (b).

Hinzu kommt eine initiale Dateiauswahl, welche in Abbildung 8 dargestellt ist.

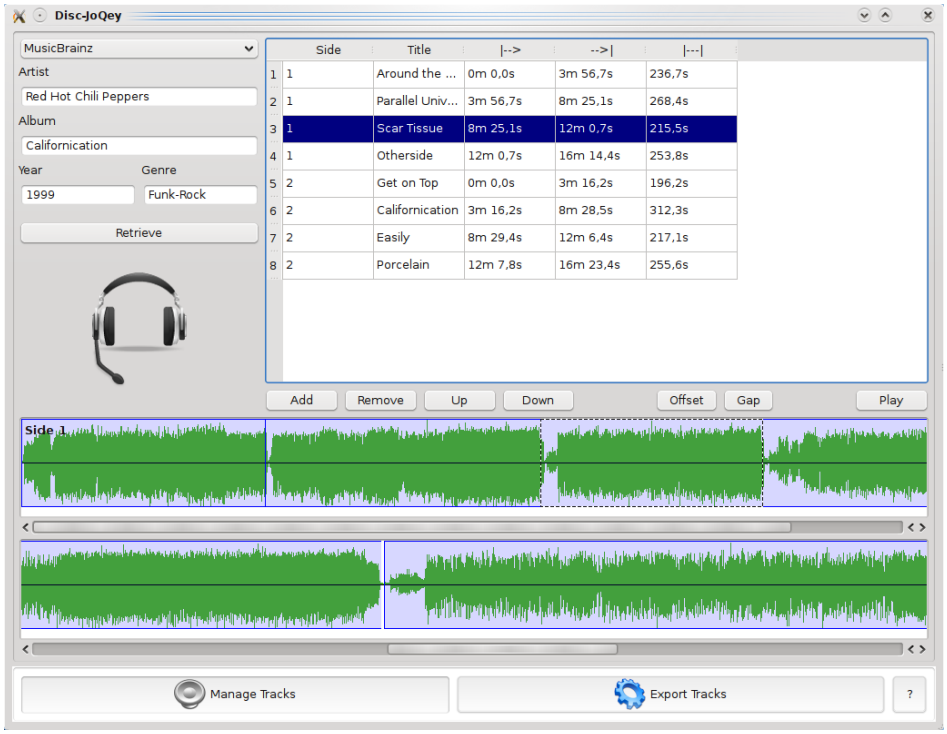
Der normale Workflow ist so angelegt, dass, nach der Auswahl der Files, zunächst über die Trackverwaltung die Tracks angelegt/verwaltet, bzw. mithilfe eines internetbasierten Metadatendienstes (konkret: Musicbrainz) ermittelt werden. Anschließend können diese Tracks über die Exportseite in MP3s umgewandelt werden.

Der Workflow ist allerdings nicht eingeschränkt, so können z.B. mehrere Verwaltungs- und Exportzyklen durchgeführt werden, oder die Tracks mehrfach (z.B. in unterschiedliche Ausgabeordner) exportiert werden.

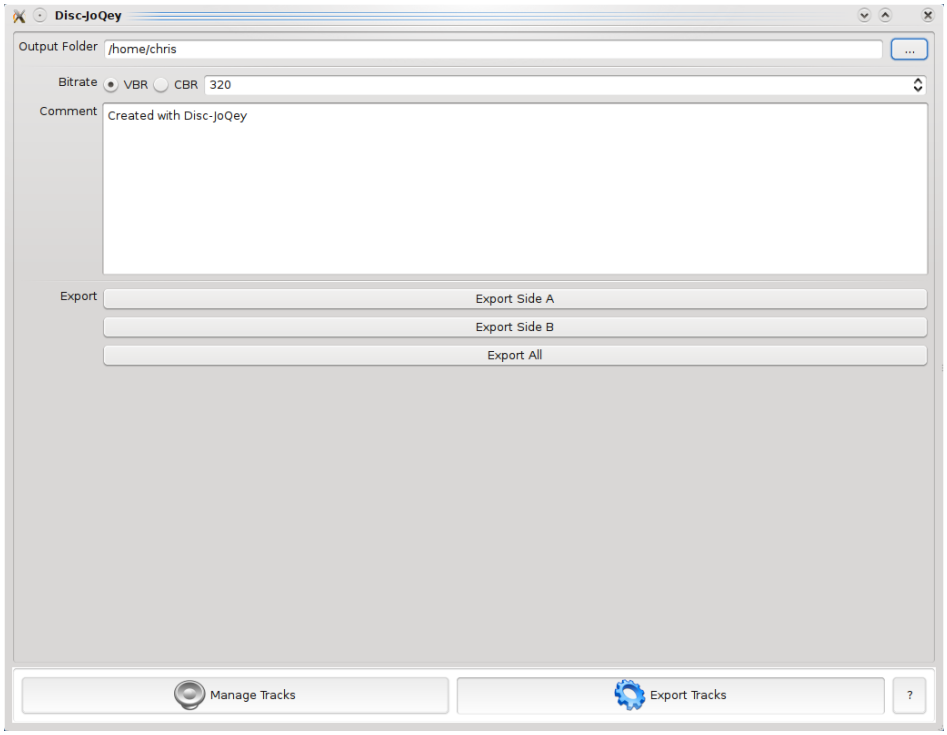
6.2. Dateiauswahl

Die Dateiauswahl ist der erste Schritt im Workflow. Hier wird zunächst ausgewählt, welche Dateien in weiterer Folge verwendet werden. Die Auswahl erfolgt mit einem Standarddialog, wie ihn der Benutzer von seiner Plattform her kennt. Mögliche Dateitypen für die Auswahl sind Wav, sowie MP3 Dateien. Wird eine MP3 Datei ausgewählt, wird sie automatisch und transparent für die weitere Bearbeitung umgewandelt.

Ferner gibt es eine Zusatzfunktion, die es dem Benutzer erlaubt, MP3 Dateien in Wav Dateien umzuwandeln. Der Benutzer kann wiederum in einem Standarddialog eine Datei zur Umwandlung auswählen und wird dann gefragt an welcher Stelle das resultierende Wav gespeichert werden soll. Ein Label informiert den Benutzer wann die Umwandlung abgeschlossen ist (“Done.”).



(a) Trackverwaltung



(b) Export

Abbildung 7: Applikation im Überblick

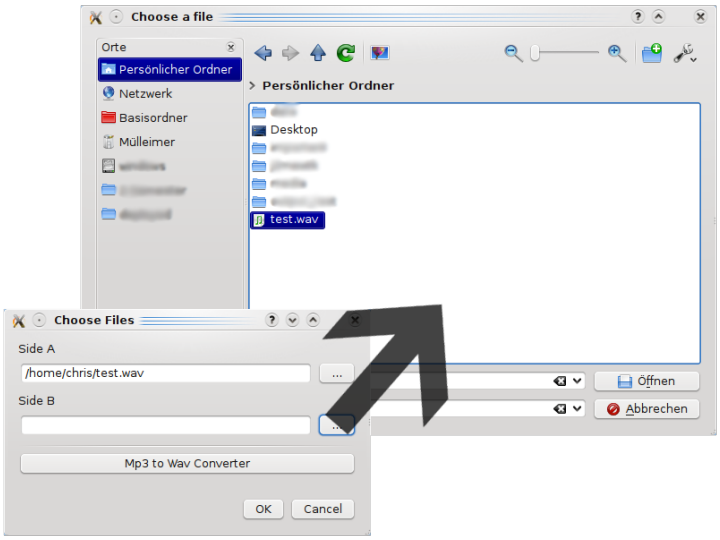


Abbildung 8: Dateiauswahl beim Applikationsstart

6.3. Trackverwaltung

6.3.1. Tracks

Mit dieser Tabelle kann die Trackliste des Albums verwaltet werden. Diese Tabelle besteht aus 5 spalten:

- *Side* Gibt die Seite an auf der sich der Titel befinden soll.
- *Title* Gibt den Namen des Stücks an.
- $|--></math> Legt den Startzeitpunkt fest.
- $--></math>| Legt den Stoppzeitpunkt fest.
- $|---|$ Zeigt die Länge des Stücks an. Eine Anpassung der Länge bewirkt eine Veränderung des Stoppzeitpunktes, der Startzeitpunkt bleibt unverändert.

Die Daten der einzelnen Felder können direkt in die Tabelle eingetragen werden. Der Wertebereich für *Side* ist dabei auf $\{1, 2\}$ beschränkt. Für Start- und Stoppzeitpunkt wird eine entsprechende Maske verwendet, um die Eingabe zu vereinfachen und Fehler auszuschließen.

Um die Trackliste zu bearbeiten werden verschiedene Funktionen angeboten:

- *Add* Fügt einen neuen Track in die Trackliste ein. Die Seite wird nach der zuletzt fokussierten Waveform gewählt.
- *Remove* Entfernt den ausgewählten Track aus der Trackliste.
- *Up* Bewegt den Track eine Position in der Trackliste nach oben.
- *Down* Bewegt den Track eine Position in der Trackliste nach unten.
- *Offset* Kann verwendet werden um für eine Seite einen globalen Offset[12] zu setzen.
- *Gap* Kann verwendet werden um für eine Seite die Dauer der Lücken zwischen den einzelnen Stücken auf einen bestimmten Wert zu setzen.

Zu beachten ist, dass die Befehle *Up* und *Down* keinen Einfluss auf die Zeiten der betroffenen Tracks nimmt. Die Trackliste ist so flexibel wie möglich ausgelegt, so können Tracks beliebig überlappend und sortiert angelegt werden, um dem Benutzer größt mögliche Freiheit zu geben.

Die Befehle *Offset* und *Gap* werden hauptsächlich für die Korrektur von Metadaten aus dem Internet verwendet.

Play bietet zusätzlich die Funktion, den aktuell ausgewählten Titel abzuspielen. Durch einen Doppelklick in die Waveform kann zu einer beliebigen Position im Track gesprungen werden.

6.3.2. Waveform

Die Waveform bietet eine graphische Repräsentation der Aufnahme (Aufbau siehe 5.2) und der eingestellten Trackliste und erlaubt die Bearbeitung dieser.

Die graphische Waveform bietet einen besseren Überblick über die Trackliste und ist wesentlicher Bestandteil, um die einzelnen Tracks per Hand zu isolieren. In der Waveform sind nämlich die Lücken zwischen den Tracks für den Benutzer einfach zu erkennen und mit der *Play* Funktion zu verifizieren.

Die Tracks werden als halbtransparente Blöcke in der Waveform dargestellt. Diese Blöcke bieten neben der Repräsentation des Tracks auch Editiermöglichkeiten. Mit der Maus

[12] Verschiebung des Ursprungs der Trackliste.

kann durch ziehen der rechten Kante die Stopzeit und durch ziehen der linken Kante die Startzeit verändert werden. Ferner ist es möglich den Block zu verschieben und so den Offset des Tracks zu veränden.

Wird der Track abgespielt, wird die aktuelle Position der Wiedergabe innerhalb des Tracks als senkrechte Linie dargestellt. Diese Position kann durch Doppelklick in den Track an eine beliebige Position gesetzt werden.

Um eine bessere Auflösung der Waveform zu erreichen, kann mithilfe des Mausrades in die Waveform "hineingezoomt" werden. Dabei wird die Waveform in die Breite gezogen und zeigt mehr Datenpunkte an.

6.3.3. Metadaten

Im Metadatenbereich lassen sich folgende Daten einstellen:

- *Artist* Der Künstler des Albums.
- *Album* Der Albumtitel.
- *Genre* Das Genre des Albums.
- *Year* Das Erscheinungsjahr des Albums.

Es werden ID3v2 Tags verwendet, was bedeutet, dass für alle Felder ein beliebiger String verwendet werden kann, so kann z.B. für das Genre ein beliebiger Name festgelegt werden, im Gegensatz zu ID3v1, wo der Wertebereich für dieses Feld auf eine bestimmte Liste von Genres limitiert ist.

Um die Trackliste nicht händisch erstellen zu müssen, gibt es die Möglichkeit die Metadaten im Internet nachzuschlagen. Zur Zeit wird als Dienst Musicbrainz unterstützt.

Als minimales Suchkriterium muss der Künstlername und der Albumtitel angegeben werden. Nach einem Klick auf *Retrieve* werden mögliche Kandidaten bei dem Metadatenservice nachgeschlagen und dem Benutzer präsentiert. Dieser kann nun den bestgeeigneten Kandidaten auswählen.

Nach Auswahl der zu verwendenden Metadaten, werden die fehlenden Metadatenfelder ergänzt, sowie die Trackliste, basierend auf den ermittelten Daten, neu aufgebaut.

An dieser Stelle ist nun meist eine Korrektur des *Offset*, sowie der *Gap*s nötig. Dies ist erforderlich, da die ermittelten Metadaten meist auf der CD-Version des Albums basieren.

6.4. Export

In diesem Schritt (Abbildung 7(b)) können nun die angelegten Tracks exportiert werden.

Dazu wird zunächst ein Ort für den Export bestimmt - wiederum erleichtert ein Standarddialog die Auswahl eines geeigneten Ordners. Anschließend wird die zu verwendende Bitrate festgelegt - näheres hierzu siehe 5.5. Das Feld "Comment" kann verwendet werden um den Metadaten der einzelnen Dateien einen optionalen Kommentar hinzuzufügen.

Abschließend kann nun mit den Export Buttons der Export gestartet werden. "Export Side X" exportiert dabei die einzelnen Seiten, "Export All" exportiert beide Seiten in einem Schritt.

7. Diskussion

Im Zuge dieses Projekts wurde eine Applikation zur einfachen Archivierung von Schallplatten entwickelt.

Die entwickelte Applikation erlaubt es dem Benutzer Schallplattenaufnahmen zu segmentieren und unterstützt ihn dabei durch eine graphische Repräsentation der Aufnahme, eine Abspielmöglichkeit zur Kontrolle, sowie der Möglichkeit Metadaten über das Internet abzufragen.

Um die großen Datenmengen effizient bearbeiten zu können, wurde eine Architektur gewählt, die Daten erst dann anfordert wenn diese auch wirklich gebraucht werden (Pipeline), sowie die Daten über einen Cache-basierten Ansatz ausschnittsweise lädt, um so einen Mittelweg zwischen Performance und Speicherverbrauch zu beschreiten.

Als effizienter Weg die Waveform der Audioaufnahme darzustellen, wurde der "Peak" (oder "Overview") Ansatz verfolgt, wie er auch in anderen populären Audioeditoren zum Einsatz kommt. Die Effizienz wird hier durch eine starke Datenreduktion - Repräsentation eines Blocks durch minimalen und maximalen Peak - erreicht.

Die segmentierte Aufnahme kann nun in MP3 Dateien exportiert werden, die mit, den eingestellten Metadaten entsprechenden, ID3 Tags versehen werden.

Die Applikation wurde als Open Source Projekt, basierend auf der BSD-Lizenz, entwickelt, was es prinzipiell jedem erlaubt die Applikation und Erkenntnisse, sofern die formulierten Bestimmungen eingehalten werden, beliebig zu erweitern und für eigene Zwecke einzusetzen.

8. Ausblick

In dieser Arbeit wurde ein halb automatischer Ansatz, basierend auf Metadaten, entwickelt, der den Menschen in den Segmentierungsvorgang mit einbindet. In weiterer Folge wären auch Ansätze denkbar, die noch weiter in die Automatisierung gehen können.

Ein Ansatz, basierend auf voll automatischer Detektion der einzelnen Tracks, ohne a priori Wissen, ist, aufgrund der in der Einleitung beschriebenen Probleme, sehr schwer umsetzbar. Was allerdings möglich wäre, ist eine Kombination aus Metadaten-basierten Ansatz und automatischer Detektion. Durch die Metadaten sind nämlich gewisse Richtwerte bekannt, an der sich die Detektion orientieren kann und so, mithilfe von weiteren Heuristiken - u.a. minimale Tracklänge, die Tracks weitgehend eigenständig segmentieren könnte.

In diesem Ansatz wäre allerdings noch immer der Nutzer in die Segmentierung involviert, da dieser erst die Metadaten überprüfen und eventuell noch bearbeiten muss. Hier allerdings könnte eine Technologie eingesetzt werden, wie sie u.a. von MusicDNS [17], ein Dienst der MusicIP Corporation, zur Verfügung gestellt wird. Dieser Dienst erlaubt es, mithilfe von Fingerprinting, Audioaufnahmen akustisch zu identifizieren und so Metadaten automatisch abzurufen.

Eine Kombination aus automatischer Detektion mit Heuristiken und Fingerprinting wäre also eine gute Möglichkeit, um die hier entwickelte Software zu erweitern und so eine voll automatische Segmentierung zu ermöglichen.

9. Literatur

[1] Living Stereo: Introducing stereophonic phonograph records; 1958. Prelinger Archives. Available from: http://www.archive.org/details/LivingSt1958.

[2] Rivait L. Forget CDs: a return to music's vinyl roots; 2009. Available from: http://pastthepages.ca/090128/arts1.html.

[3] The Linux Information Project. BSD License Definition; 2005. Available from: http://www.linfo.org/bsdlicense.html.

[4] Bezemer A. GramoFile; 2001. Available from: http://www.opensourcepartners.nl/~costar/gramofile/.

[5] AlpineSoft. VinylStudio;. Available from: http://www.alpinesoft.co.uk/.

[6] Nokia. Qt Reference Documentation; 2009. Available from: http://doc.qtsoftware.com/4.5/index.html.

[7] Erik de Castro Lopo. Libsndfile; 2009. Available from: http://www.mega-nerd.com/libsndfile/.

[8] Projektwiki. MusicBrainz; 2009. Available from: http://wiki.musicbrainz.org/MusicBrainz.

[9] Projektwiki. Libmusicbrainz; 2009. Available from: http://wiki.musicbrainz.org/libmusicbrainz.

[10] The LAME Project. Projekt - Homepage; 2008. Available from: http://lame.sourceforge.net/.

[11] id3lib. The ID3v1/ID3v2 Tagging Library; 2003. Available from: http://id3lib.sourceforge.net/.

[12] Ibanez L, Schroeder W, Ng L, Cates J. The ITK Software Guide, Second Edition; 2005. Available from: http://www.itk.org/ItkSoftwareGuide.pdf.

[13] Handy J. The Cache Memory Book: The Authoritative Reference on Cache Design. vol. 2. Morgan Kaufmann; 1998.

[14] Trolltech and KDE Cooperate on Cross-Platform Multimedia Programming Framework.; 2007. Available from: http://www.qtsoftware.com/about/news/archive/press.2007-12-11.2263733764.

[15] KDE. Using Phonon; 2009. Available from: http://phonon.kde.org/cms/1022.

[16] Betz J. MP3. Pearson Education; 2004.

[17] MusicIP Corporation. MusicDNS; 2009. Available from: http://www.musicip.com/dns/index.jsp.

Bilderquellen

Die Bilder wurden, sofern nicht anders angegeben, vom Autor selbst erstellt.

A. BSD Lizenz

Es folgt die verwendete Lizenz in ihrer Gänze. Sie ist gültig für das gesamte Projekt, inklusive dieses Whitepapers: