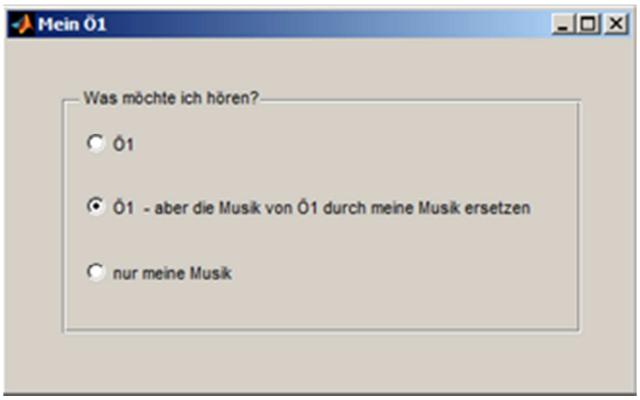
Gui and Usage

Start des Programms über Eingabe von „gui“ im Matlab Command Window. Es erscheint zunächst ein Waitbar, da die Anwendung beim Start 20 Sekunden des ö1 Streams buffert. Danach wird dann nur noch das Graphical User Interface der Anwendung angezeigt:



Graphical User Interface von MeinÖ1

Das User Interface bietet drei Auswahlmöglichkeiten:

- Ö1
- Ö1 – aber die Musik von Ö1 durch meine Musik ersetzen
- nur meine Musik

Die erste Option gibt den Ö1 Live Stream ohne eine Veränderung wieder.

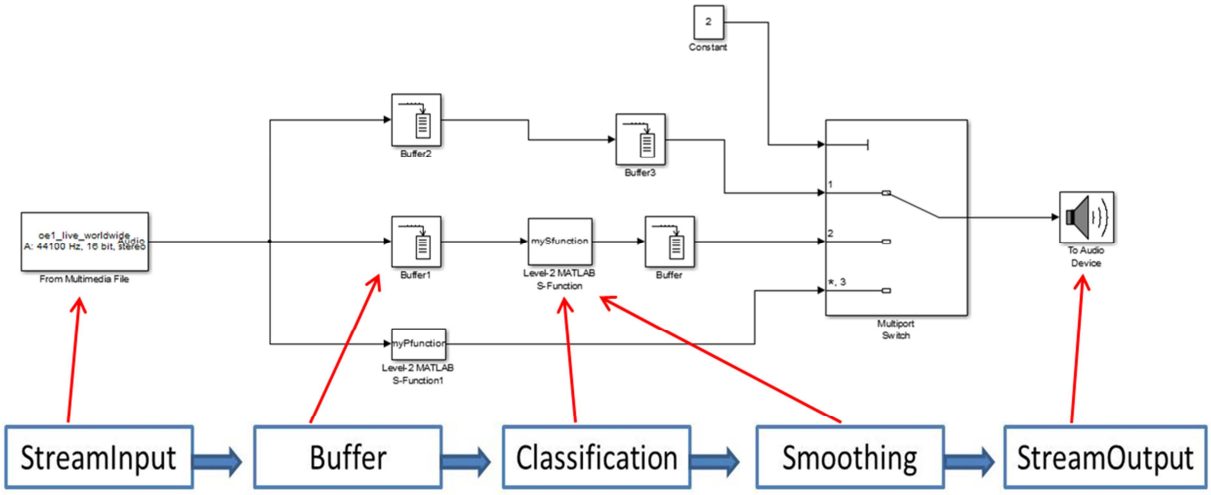
Die zweite Option ersetzt die Musikstücke von Ö1 durch die Songs, die der User im Ordner „meineMusik“ gespeichert hat. (Derzeit sind im Ordner „meineMusik“ ca. 60 Reggae Songs gespeichert, diese vielleicht für das Probehören durch eigene Files ersetzen. Die Anwendung kann folgende Dateiformate verarbeiten: wav, wma, aif, aifc, aiff, mp3, au, snd, mp4, m4a, flac, ogg) Beim Start der Anwendung wird eine Liste der Files im Ordner „meineMusik“ geladen und in eine zufällige Reihenfolge gebracht. Diese Zufallsreihenfolge wird dann verwendet um die Songs von Ö1 zu ersetzen. Sind alle Songs aus dieser ersten Zufallsreihenfolge abgespielt wird eine neue Zufallsreihenfolge erstellt.

Ist die dritte Option ausgewählt werden nur die Songs abgespielt, die im Ordner „meineMusik“ gespeichert sind.

Streaming - Verarbeitungskette

Ist die zweite Option am GUI ausgewählt(Ö1 – aber die Musik von Ö1 durch meine Musik ersetzen) wird folgende Verarbeitungskette von der Anwendung abgearbeitet:

Für diese Verarbeitungskette verwendet „MeinÖ1“ ein Matlab Simulink Model(oe1Model.slx):



Simulink Model und Streaming - Verarbeitungskette

StreamInput

Ein „FromMultimediaFile“ Block aus der Simulink DSP Systemtoolbox nimmt den Ö1 Live Stream von der Adresse mms://apasf.apa.at/oe1_live_worldwide auf.

Buffer

Die Anwendung buffert den Input Stream in 20 Sekunden Chunks. Diese Chunks werden an die nachfolgende Verarbeitungsschritte weitergegeben. Der nachfolgende Level-2 MATLAB S-Function Block übernimmt den Chunk und ruft die Classification und Smoothing Stages der Verarbeitungskette auf.

Classification

Der Code für das Classification Stage befindet sich in der Funktion “functions/classifyAudioStream.m“

Im Classification Stage der Anwendung wird aus jedem 20 Sekunden Chunk zunächst mit 0.04 sec Short Term Windows(non overlapping) 35 Short Term Features extrahiert: Energy, Zero crossing rate, Entropy of energy, Spectral centroid, Spectral spread, Spectral entropy, Spectral flux, Spectral rolloff, Mel-Frequency cepstrum coefficients, Chroma vector, Harmonic ratio, Fundamental frequency.

Im nächsten Schritt wird aus jedem Short Term Feature mit 1 sec Mid Term Windows(0,75 sec overlapping) einige statistische Lagemasse berechnet: Durchschnitt, Median, Standartabweichung, Standartabweichung des Durchschnitts, Maximum, Minimum.

Insgesamt ergibt sich ein 210 dimensionaler Feature-Vektor pro Mid Term Window. Der in WEKA trainierte Classifier (siehe Abschnitt „Classifier Training“) klassifiziert jedes Mid Term Window anhand dieses Feature-Vektors.

Im letzten Schritt des Classification Stage werden alle gleich klassifizierten Mid Term Windows zu Segmenten zusammengefasst. Eine Liste mit den Startzeitpunkten und den Klassenlabels („Musik“, „Sprache“, „ Stille“) der gleich klassifizierten Segmente wird an das Smoothing Stage weitergegeben.

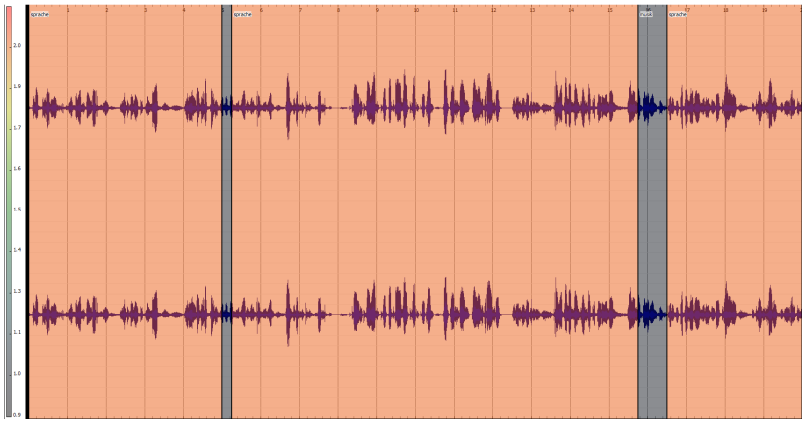
Smoothing

Der Code für das Smoothing Stage befindet sich in der Funktion “functions/smoothing.m“

Im Smoothing Stage der Anwendung werden die Fehler des Classifiers behoben. Es folgt eine Beschreibung der häufigsten Fehlerfälle.

Kurze Fehler in Sprach Chunks

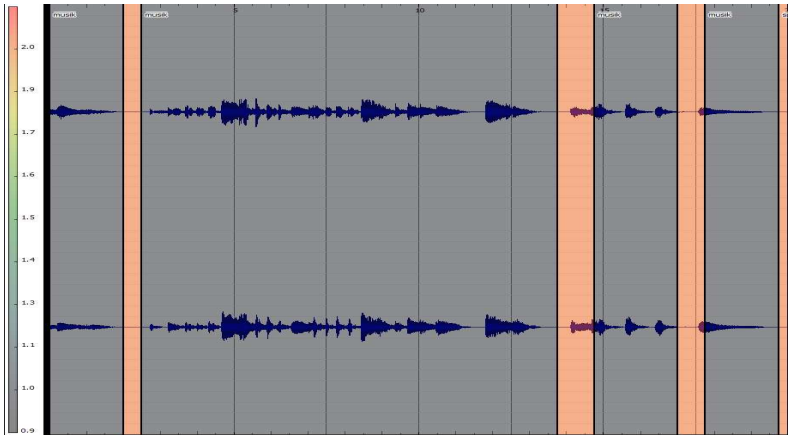
Der Chunk besteht zu 100% Prozent aus einer Sprachaufnahme. Trotzdem werden kurze Segmente (0,25 -1,25 sec Dauer) als Musik klassifiziert. Der Fehler tritt sehr häufig bei ausgebildeten Sprechern auf, die sehr ausgeprägte Vokale bilden, bei Telefoninterviews und Interviews mit vielen Hintergrundgeräuschen auf.



20 sec Sprach Chunk mit kurzen Fehlklassifikationen

Kurze Fehler in Musik Chunks

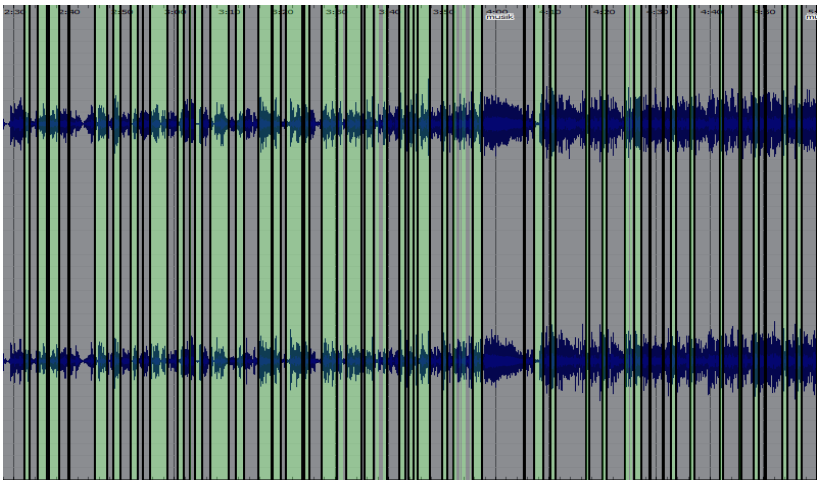
Der Chunks besteht zu 100% Prozent aus Musik. Vom Classifier werden aber kurze Segmente(0,25 - 1,25 sec Dauer) als Sprache klassifiziert. Tritt häufig in der Attackphase von Tönen auf.



20 sec Musik Chunk mit kurzen Fehlklassifikationen

Musik, die den Classifier total verwirrt

Bei Musikstücken wie z.B. Gedichtvorträge mit leiser Musik, Experimentalmusik in der viele Geräusche verarbeitet werden, Arien ohne Orchesterhintergrund, usw. wechselt der Classifier in rascher Abfolge zwischen Sprach- und Musiksegmenten hin und her. Derartige Chunks werden zu reinen Musik Chunks geglättet.



3 min Musik, die den Classifier total verwirrt

Mixed Chunks

Kurze Musikeinspielungen in Reportage Beiträgen und die Kennmelodien von Sendungen werden vom Classifier als Musik klassifiziert, sollen aber nicht durch Musik des Users ersetzt werden.

Derzeit ist die Mixed Chunk Logik darauf programmiert, dass Reportage Beiträge zur Gänze zu hören sind und die kurzen Musikeinspielungen in diesen Beträgen nicht durch die Musik des Users ersetzt werden. Diese Logik führt aber manchmal dazu dass die Anfangstöne von Songs zu hören sind bevor zur Musik des Users gewechselt wird. Während der Entwicklung wurde Wert darauf gelegt dass Reportage Beiträge zu Gänze zu hören sind und die manchmal auftreten Verzögerungen beim Umschalten auf die Musik des Users als dafür notwendiges Übel in Kauf genommen. Grundsätzlich wäre es möglich jede kleinste Musikeinspielung durch die Musik des Users zu ersetzen was aber das Hörerlebnis des Users bei Reportage Beträgen wesentlich beeinträchtigt.

Mixed Chunk: kurze Musikeinspielung in einem Beitrag

StreamOutput

Ein „ToAudioDevice“ Block aus der Simulink DSP Systemtoolbox reicht den klassifizierten und geglätteten Chunk an die Soundkarte des PCs weiter.

Classifier Training

Die für das Classifier Training verwendeten Soundfiles sind im Ordner „radioSamples/classifier TrainingMaterial“ zu finden. Aus diesen Soundfiles wurde durch das Script „saveARFFfileForClassifierTraining.m“ die Features extrahiert und ein arff File erzeugt. Die im Laufe der Entwicklung erzeugten arff Files wurden im Ordner „arff_files“ abgespeichert. Mit diesem arff File wurde in Weka 3.6 der Classifier trainiert. Die besten Kennzahlen erreicht die in Weka implementierte Support Vector Maschine(weka.classifiers.functions.SMO):

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       29268               99.7002 %
Incorrectly Classified Instances        88                0.2998 %
Kappa statistic                          0.994
Mean absolute error                      0.2229
Root mean squared error                  0.2734
Relative absolute error                 66.4495 %
Root relative squared error             66.7601 %
Total Number of Instances            29356

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.999     0.005      0.996     0.999     0.997      0.997    musik
                 0.994     0.001      0.999     0.994     0.997      0.997    sprache
                 1         0          0.993     1         0.996      1        stille
Weighted Avg.    0.997     0.003      0.997     0.997     0.997      0.997

=== Confusion Matrix ===

     a     b     c   <-- classified as
 16136    14     0 |     a = musik
    72 12860     2 |     b = sprache
     0     0   272 |     c = stille
```

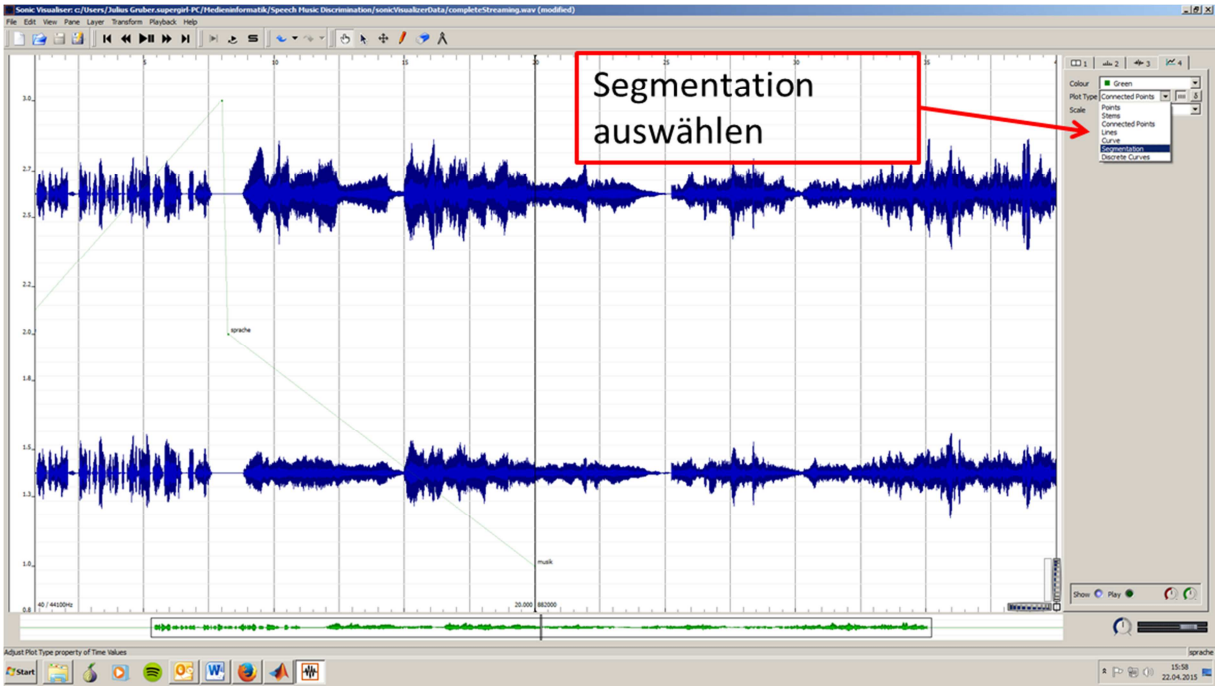
Visualisierung

Während der Entwicklung wurde Sonic Visualizer zur Visualisierung und Analyse der Classifier und Smooting Stages verwendet. Sonic Visualizer kann unter folgendem Link heruntergeladen werden: http://www.sonicvisualiser.org/download.html

Die Anwendung protokolliert für jeden verarbeiten Chunk die Ergebnisse des Classifiers und Smoothing Stages. Im Ordner „sonicVisualizerData“ werden pro Chunk ein wav File und zwei Text Files abgespeichert. Die wav Files sind *„streamedSingleChunk[chunkNumber].wav“* benannt. Die Ergebnisse des Classifier Stages werden in den Text Files mit der Benennung *„streamedSingleChunkBeforeSmoothing[chunkNumber].txt“* abgespeichert. Die Ergebnisse des Smoothing Stages werden in den Text Files mit der Benennung *„streamedSingleChunk[chunkNumber].txt“* abgespeichert.

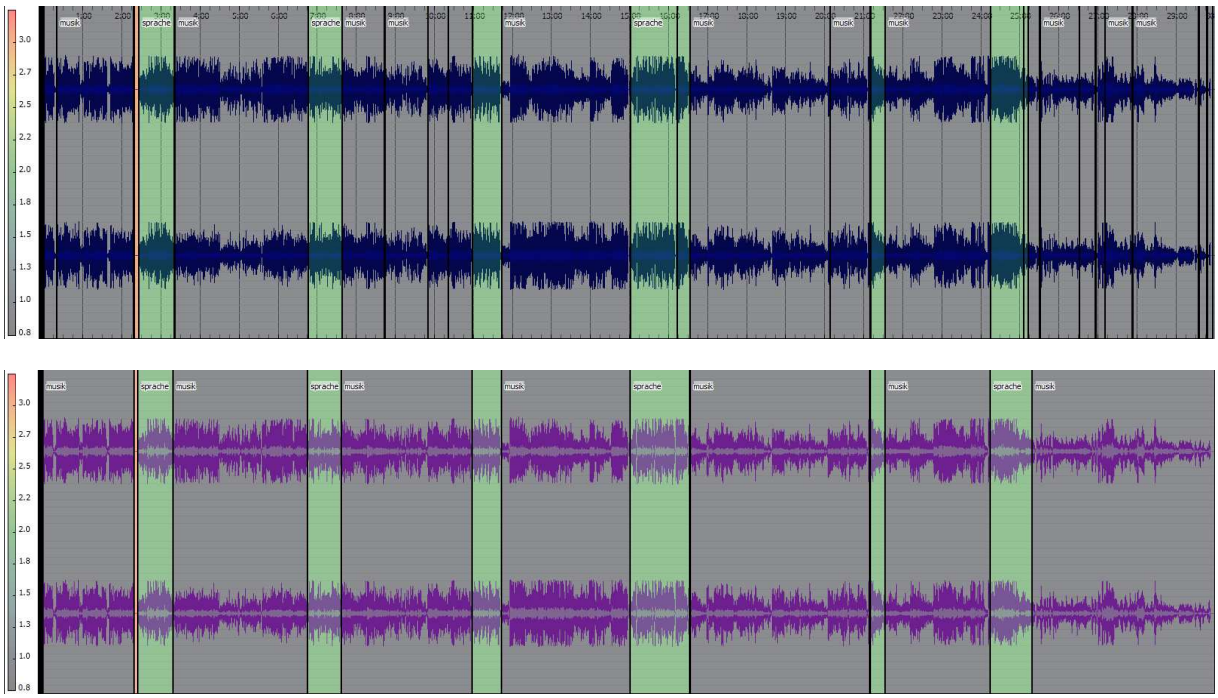
Beim Beenden von „MeinÖ1“ wird ein wav File für die gesamte Zeit die „MeinÖ1“ verwendet wurde erstellt(*„completeStreaming.wav“*). Die Ergebnisse des Classifier und Smoothing Stages werden in den Text Files *„completeStreamingBeforeSmoothing.txt“* bzw. *„completeStreaming.txt“* gespeichert.

Diese Files können in Sonic Visualizer für die Visualsierung verwendet werden. Im Menü „File“ befinden sich die Befehle „Open…“ für das Öffnen der wav Files bzw. „Import Annotation Layer…“ für das Öffnen der Text Files. Danach links als „Plot Typ“ Segmentation auswählen.



Sonic Visualizer nach dem Öffnen des Wav Files und Import des Text Files

Nach der Auswahl des Plot Typs zeigt Sonic Visualizer die von Mein Ö1 erzeugte Segmentierung an:



30 min MeinÖ1 Ouput, Oben: Classifier Output, Unten: Smoothing Output

Known Problems and Future Work

Ein wesentliches Problem sind die Kennmelodien für bestimmte Sendungen. Diese werden vom Classifier als „Musik“ klassifiziert und werden damit durch die Musik des Users ersetzt. Der Versuch dieses Problem im Smoothing Stage der Anwendung zu beheben gelingt nur teilweise. Eine Verbesserung könnte eine Audio – Identifikation bringen: Ein vom Classifier als „Musik“

klassifiziertes Chunksegment könnte in einem weiteren Verarbeitungsschritt dahingehen überprüft werden ob es sich um eine Kennmelodie einer Sendung handelt.

Wechselt der User am GUI die Klangquelle dauert es ca. 2-3 Sekunden bis die Klangquelle tatsächlich wechselt. Hier gibt es einen Tradeoff zwischen dem raschen Umschalten auf eine andere Klangquelle und kurzen Tonausfällen im Audiostreaming. MeinÖ1 ist derzeit so konfiguriert das es zu keinen Tonaussetzern kommen sollte, was aber das Umschalten auf eine andere Klangquelle etwas träge macht.

Die Übergänge zwischen den Sprachbeiträgen von Ö1 und der Musik des Users sind derzeit harte Cuts ohne FadeOuts und FadeIns. Das Wechseln klingt daher manchmal etwas abrupt.

Eine weitere Verbesserung des Smoothing Stages könnte die Einbeziehung des Sendeschemas von Ö1 bringen. Auf Musiksendungen könnte ein anderer Smoothing Algorithmus angewandt werden als auf Sendungen die Nachrichten und Reportagen bringen.

Größere Chunks zu verwenden würde die Smoothing Logik ebenfalls verbessern. Würde die Anwendung mit z.B. 1 Minuten langen Chunks arbeiten könnte jedes Musik Segment unter einer Dauer von 30 Sekunden als Signation, Musikeinspielung während eines Reportage Betrags, usw. erkannt werden. Diese Option wurde in Hinblick auf den Livecharakter von Radio nicht verwirklicht: Wenn der Nachrichtensprecher sagt „ Es ist 12 Uhr“ soll die Anwendung nur wenige Sekunden hinter der tatsächlichen Uhrzeit liegen.