Uppgift 1

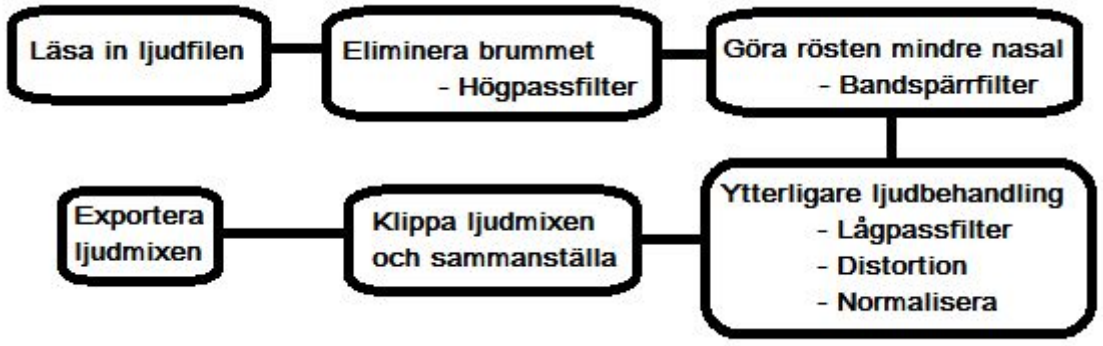
En av grupperna som gör filmprojekt i kursen TNM088 – Digitala medier har precis gjort klart sin videoredigering och är väldigt nöjda med resultatet. Det är bara ett problem; då filmberättelsen är tämligen komplicerad har de valt att använda en berättarröst (en voice-over) för att förenkla berättelsen för publiken. Inspelningen och redigeringen av berättarrösten gjordes dock till en tidig version av filmutkastet, och originalinspelningen försvann i en hårddiskkrasch, så nu stämmer inte tajmingen mellan berättarröst och bild längre. Dessvärre har inspelningen även ett lågfrekvent brum från kylskåpet i köket där inspelningen gjordes, och en dålig mikrofon gör att berättarrösten låter nasal och lite gäll i det övre mellanregistret. Tiden innan redovisningen räcker inte till för att spela in en ny version, utan uppgiften är att skapa en version som har rätt tajming och bra ljud utifrån den befintliga ljudfilen, beskriv hur ni skulle gå till väga.

2017-10-25

Hemtenta, uppgift 1

Först tittar jag på filmen, sedan lyssnar jag på den redigerade inspelningen för att få en överblick över vad jag har att arbeta med. Sedan ser jag på filmen igen och skriver samtidigt ner på papper vid vilka tidpunkter specifika lines av berättarrösten ska komma in och hur många sekunder dessa berättarsekvenser varar. Efter det lyssnar jag på inspelningen igen och skriver ner alla lines som ska vara med i filmen och vid vilken tidpunkt de ska starta i filmen, och hur långa dom är.

All behandling av ljudet kommer ske inom programvaran Matlab. Scriptet som jag kommer skriva kommer följa strukturen; först fixas ljudkvalitén och sedan klipper jag om inspelningen. En mer detaljerad pseudokod över hela scriptet kan ses i figur 1.



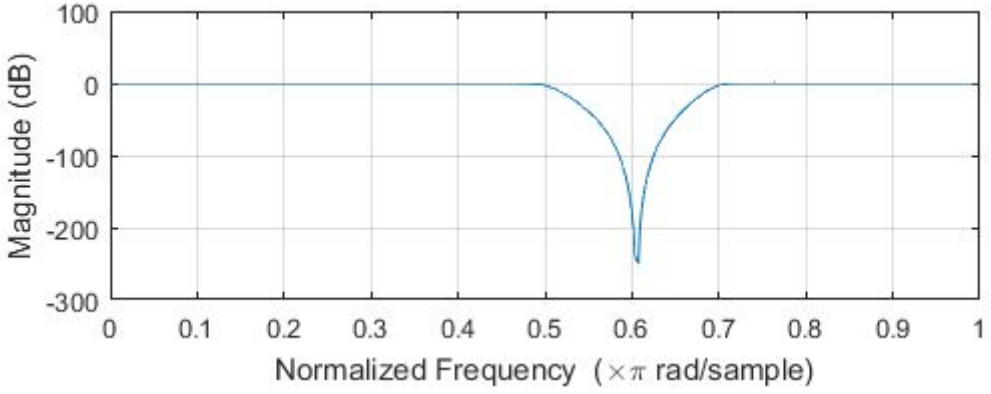
Figur 1 - Pseudokod för scriptet.

Jag börjar med att läsa in filen med den inbyggda Matlab-funktionen *"audioread"* och får därmed ut en hela inspelningen som en vektor och dess samplingsfrekvens.

Det lågfrekventa brum från kylskåpet som hörs i inspelningen ska bort. Först öppnar jag ett *"spectrogram"* i Matlab, för att identifiera brummet som en gul horisontell linje i spectrogrammet. Det går att välja att visa de lägre frekvenserna mer inzoomat och därmed kan jag ungefärligt identifiera frekvensen för brummet. För att eliminera brummet använder jag ett *högpassfilter* för att radera alla frekvenser under en brytfrekvens som jag själv bestämmer. Denna brytfrekvens väljer jag en liten bit ovanför frekvensen för brummet. Det gör jag för jag vill undvika en resonans som kan uppstå vid frekvenser ovanför brytfrekvensen[1] och därmed förstärka eventuella frekvenser för det brummande ljudet. Jag måste även vara noggrann att inte ha brytfrekvensen över fundamentalfrekvensen för mänskligt tal. Sedan filtrerar jag ljudet och för att kontrollera att brummet är borta kan jag

antingen lyssna på resultatet, eller plotta ett nytt spectrogram och titta ifall den gula linjen finns kvar. Användningen av högpassfilter är inte bara bra för att få bort brummet, men även bra för att ta bort eventuella onödiga låga frekvenser som smugits in under inspelning som kan ha missats. Konventionellt sett brukar sådana oönskade frekvenser tas bort inom mastering[2]

Jag måste även åtgärda att berättarrösten låter nasal och gäll i det övre mellanregistret. För att konfirmera att det stämmer använder jag *"fft"* dvs. en fast Fourier transform (FFT), vilket är en effektiv algoritm för att beräkna en diskret, begränsad fouriertransform. Denna använder jag för att plotta inspelningen i frekvensdomänen för att kunna se hur mycket av signalen som ligger inom varje givet frekvensband över ett intervall av frekvenser.[3] Med denna plot har jag både konfirmerat att inspelningen har höga frekvenser i det övre mellanregistret och jag har lyckats identifiera vilket frekvensområde i mellanregistret som orsakar att rösten blir nasal och gäll. Nu vill jag sänka nivån på detta frekvensområde. Jag väljer att inte använda ett lågpassfilter eftersom det filtret skulle eliminera alla frekvenser över en viss brytfrekvens. Jag vill behålla den dynamiska omfången av berättarrösten så mycket som möjligt och jag vill inte riskera att det skulle låta som berättaren satt i en låda och spelade in, vilket kan ske om för mycket av de övre frekvenser tas bort och det dynamiska omfånget är för litet. Istället använder jag ett bandspärrfilter/bandstoppfilter som dämpar ett specifikt område av frekvenser, men släpper igenom alla andra frekvenser.[4] I Matlab finns ett sådant filter i funktionen "Butterworth" där en order av sju, ger ett filter som ser ut så här, se figur 2. Det är viktigt att tänka på är att funktionen vill ha en övre och lägre brytfrekvens som ska matas in som normaliserade frekvenser, och dessa anger vilket frekvensområde som ska dämpas.



Figur 2 - Exempel på hur bandspärrfiltret kan se ut

För att ta bort ytterligare onödiga frekvenser som inte behövs i mixen använder jag ett lågpassfilter, som tar bort frekvenser ovanför brytfrekvensen[5] , där brytfrekvensen är ca. 7000 Hz. Det är för frekvensomfånget för tal inte överstiger 7000 Hz[6] och därför gör det inget att jag tar bort alla frekvenser över det.

Nu ska jag normalisera ljudnivån genom en lätt distortion och jag ska vara noggrann med att inte tillföra för mycket distortion med risk för att förstöra ljudkvalitén. Syftet med att använda distortion är att både öka hörbarheten en aning och komprimera dynamiken en aning för att få en mer jämn ljudnivå genom mixen. Grundtanken för distortion är att ljudvågorna ska bli mer lika fyrkantsvågor och därmed förstärka övertonerna.[7] Detta kan göras genom att komponentvis multiplicera ljudvektorn med en faktor, som bestämmer förstärkningen, och sedan komponentvis dividera detta med siffran ett adderat med absolutbeloppet av ljudvektorn multiplicerat med förstärkningsfaktor.

Det är viktigt att kontrollera att ljudfilen inte klipper för då låter ljudet dåligt och ansträngt. Jag plottar ljudet genom att använda *“plot”* i Matlab för att se ifall ljudet klipper. Sedan dividerar jag ljudvektorn med maxvärdet från ljudvektorn.

Nu är det dags att “klippa” själva ljudet och skapa den slutgiltiga ljudmixen som ska användas i filmen. Först vill jag skapa en tom vektor så att den matchar längden av filmen. Eftersom jag i Matlab arbetar med sampel, dvs. att vektorerna innehåller ett sampel på varje position, måste jag ta reda på hur många sampel som motsvarar en sekund för att kunna matcha vektorns storlek med längden av filmen. Eftersom samplingsfrekvensen är antalet sampel dividerat med tiden i sekunder, får jag genom att multiplicera längden av filmen (i sekunder) med samplingsfrekvensen, ut hur många sampel vektorn för hela ljudmixen måste ha. Genom att använda samma formel men sätta antalet sekunder som värdet ett, får jag även ut att antal sampel per sekund är lika med samplingsfrekvensen vilket kan ses som trivialt.

Nu ska jag spara varje dialogsnutt i separata vektorer som är döpta efter tidpunkten de ska spelas upp i filmen, tex. *“line_02_23”* är en line som ska spelas upp vid tidpunkten *2.23* i filmen. Jag använder de tidigare anteckningarna jag gjorde i början till detta. Jag skriver en funktion som fungerar som i figur 3.

Figur 3 - Pseudokod för funktionen

Funktionen returnerar en vektor med en specifik berättarsekvens som jag själv väljer. Som indata tar funktionen start- och sluttid för sekvensen, och den behandlade ljudmixen, och som utdata returnerar den en vektor som enbart innehåller just den valda dialogen men som har samma längd som hela ljudmixen för hela filmen. Funktionen använder sig av att en sekund, som tidigare nämnt, är lika med samplingsfrekvensen, så detta används till att ta reda på vilken del av ljudmixen som den specifika dialogen befinner sig. Detta görs genom att multiplicera start-och sluttiden med samplingsfrekvensen, och dessa används som index till att bevara berättarsekvensen i ljudmixen, och sätta resten av positionerna i ljudmixen till värdet noll. Det görs genom att skapa en vektor med ettor i ett intervall som är lika långt som dialogen. Sedan används *"padding"* för att sätta dit nollor både före och efter ettorna så att den vektorn blir lika lång som vektorn för hela ljudmixen är. Slutligen multipliceras vektorn som enbart innehåller nollor och ettor med vektorn, med själva ljudmixen, och då får jag en vektor som enbart innehåller dialogen. Funktionen returnerar denna vektor. Alla returnerade enskilda dialoger summeras ihop till en vektor som är den slutgiltiga ljudmixen för hela filmen. En sista kontroll görs för att kontrollera att ljudmixen inte klipper. Sedan exporteras ljudmixen till en fil med den inbyggda Matlabfunktionen *"audiowrite"*, och nu är den redo för att användas till filmen.

Referenser

[1] Acoustica, https://www.acoustica.com/mixcraft/help/mixc/low_pass_cutoff_low_pass_resonance.htm, *Hämtad 2017-10-25*

[2] Mathworks, https://se.mathworks.com/discovery/high-pass-filter.html, *Hämtad 2017-10-25*

[3] Matworks, https://se.mathworks.com/help/matlab/math/fourier-transforms.html, *Hämtad 2017-10-25*

[4] All about circuits, https://www.allaboutcircuits.com/textbook/alternating-current/chpt-8/band-stop-filters/, *Hämtad 2017-10-25*

[5] Mathworks, https://se.mathworks.com/discovery/low-pass-filter.html, *Hämtad 2017-10-25*

[6] AV Info, http://www.bnoack.com/index.html?http&&&www.bnoack.com/audio/speech-level.html, *Hämtad 2017-10-25*

[7] Case, A. (2012). *Sound FX: Unlocking the Creative Potential of Recording Studio Effects*, https://books.google.com/books?id=sfL7AwAAQBAJ&pg=PA93&lpg=PA93&dq=Severely+clipping+a+sine+wave+until+it+resembles+a+square+wave+has+the+effect+of+adding+harmonic+&source=bl&ots=_qmKig2oq0&sig=ZvGWRyM9ONIJRXaaDVMS-RAYky0&hl=sv. *Hämtad 2017-10-25*