Vocol Harum

Generering av harmonisk treklang

A. Lundin *alilu684*, E. Sandrén *erila135*, K. Eskilson *klaes950*

TFYA65 – Ljudfysik

10 oktober 2014

Sammanfattning

Denna rapport redogör för ett projektarbete gjort i kursen TFYA65 – Ljudfysik. Projektets syfte är att med hjälp av fysikaliska modeller samt tillämpningar lära sig mer inom ljud- och signalbehandling.

Innehåll

1 Bakgrund

Projektets mål är att skapa en så kallad vocal harmonizer. Detta innebär ett program som lägger en ters och en kvint ovanpå en ton, och därigenom skapar en treklang på den ton som mottages av programmet.

1.1 Diatoniska intervall

Hädan efter krävs en förståelse för vad en skala innebär, och vad diatoniska intervall är.

Ters och kvint är förhöjningar av frekvensnivån utav grundtonen enligt *tabell 1*. För att nå kvint/ters behöver alltså originalljudets frekvens höjas. Då dessa värden är helt oberoende av ton är det irrelevant att ta reda på grundtonens egna frekvens för att göra frekvenshöjningen [2]. Enbart en av liten och stor ters är aktuell i en skala, och hädanefter kommer den stora tersen att användas.

Diatoniskt intervall	Frekvensfaktor	Exempel (hz)
Grundton	1	440
Liten ters (moll)	1.2	528
Stor ters (dur)	1.25	550
Kvint (dur och moll)	1.5	660

Tabell 1: Diatoniska intervall

2 Metod

Samtliga medlemmar i gruppen började arbetet med att läsa på om signalbehandling och tidigare experiment som gjorts inom området. Arbetet delades sedan upp i mindre projekt som sammanfördes och utvärderades i slutet av varje arbetspass, och det bestämdes vad som kunde sparas och inte.

Det verktyg som använts främst i projektet är MATLAB. För inspelning och uppspelning användes datorns inbyggda hårdvarukomponenter.

Ett program skapades som läser in en signal och gör om den till frekvensdomän med hjälp av inbyggda transformfunktioner i MATLAB. Det görs då ljudets fasgång inte går att förändra i tidsdomänen. Programmet delar upp ljudet i mindre segment där brus reduceras, och fasgången studeras. Skillnaden i fasgång mellan nuvarande segment och förgående beräknas [1, kapitel

7.4]. Genom denna skillnad kan en syntetisk fasgång skapas. Den syntetiska fasgången måste användas för att undvika att ett fasfel uppstår vid omsamplingen [4]. Signalen transformeras sedan tillbaka till tidsdomänen med hjälp av invers fouriertransform, se *programkod 1*. Segmentet samplas om i tidsdomänen genom att samplingsvärden med visst intervall raderas. Intervallet är relativt till vald tonhöjning. En linjär interpolation sker av kvarstående samplingsvärden för att ljudet skall vara kontinuerligt. De nya tonerna, kvinten och tersen, kan sedan adderas med grundtonen i tidsdomänen.

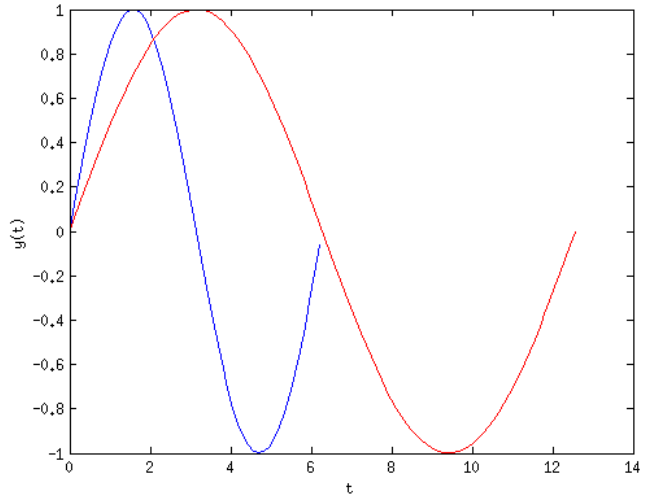
Programkod 1: fouriertransform (fft) samt invers fouriertransform (ifft)

```
f         = fft(fftshift(grain));
r         = abs(f);
meanValue = mean(r);
threshold = meanValue*1.1;
r(r < threshold) = 0;
grain = fftshift(real(ifft(f)));
```

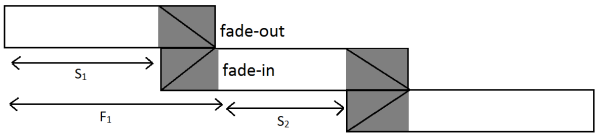
För att reducera brus görs åtgärder på två sätt. Medelvärdet av frekvensamplituden i varje segment beräknas och multipliceras med 1,1. Detta värde verkar som tröskelvärde, och alla frekvenser inom tidsintervallet med amplitud lägre än tröskelvärdet filtreras bort. Detta fungerar bra i de segment där någon ton är dominant, men vid inspelning med pauser (exempelvis en inspelning av sång) kommer vissa segment ha medelintensitet som är för låg för att verka som tröskelvärde. Det skulle resultera i att bruset reduceras då en stark ton spelas och sedan förstärks vid tystnad.

Den andra åtgärden kontrollerar därför medelvärdet av ljudintensiteten i varje segment. Är detta värde lägre än 30dB modifieras inget ljud alls och programmet går vidare till nästa segment.

För att höja frekvensen på ett ljud flyttas de sampelpunkter som verkar på originalsignalen isär med en önskad faktor. De trycks sedan ihop till ursprungligt avstånd och signalen "följer med", se *figur 1*. På så sätt fås ett tidsfel som beror på faktorn för samplingen, det vill säga den grad som frekvensen ska höjas. Under uppspelning av ljud blir därför kvintens ljud något kortare än originalet och tersen ännu kortare. En lösning på problemet är repetition och överlappning [3]. Signalen delas upp i hanningfönster. För att förlänga signalen repeteras skärningsdelen av angränsande segment, se *figur 2*. Det ger visserligen ett fel, men om segmenten är tillräckligt små och repetitionsdelen liten kommer detta inte att höras.



Figur 1: *Godtycklig sinussignal i tidsdomänen. Originalsignalen (röd) har förhöjts till dubbla frekvensen och därmed tryckts ihop (blå).*



Figur 2: *Överlappning av fönster, F, gör att mellanrummet mellan segment S1 och S2 ökas.*

En metod för att använda funktionen i realtid gjordes till sist genom att dela upp inspelningen i korta intervall om cirka en halv sekund. Då programmet spelat in ljud från ett tidsintervall får det genast genomgå transformation och spelas sedan upp i högtalaren samtidigt som nästa tidsintervall spelas in. För att simulera realtid görs sampeltiden liten.

3 Resultat

Det slutgiltiga resultatet är en MATLAB-funktion som behandlar ett förinspelat ljudstycke. Ljudet läses och delas upp i segment på 512 sampels. Dessa segment studeras i hanningfönstren på 2048 samples. Ljudintensiteten i varje segment kontrolleras mot ett tröskelvärde och om det överstiger värdet transformeras ljudsegmentet om till frekvensdomänen med Fouriertransform. Brus reduceras och en fasgenerering görs. Signalen transformeras

tillbaka med invers Fouriertransform. Frekvensen samplas sedan om med rätt faktor för att skapa ters och kvint. De nya tonerna adderas till originaltonen, och nästa segment behandlas. Om intensiteten inte överskrider tröskelvärdet förblir segmentets värden oförändrade, och programmet går vidare till nästa segment. Då samtliga tidsintervall transformerats och omsamplats spelas den nya melodin upp.

4 Diskussion

I och med att MATLAB ej är gjort för snabb och effektiv beräkning i realtid inleddes arbetet med att göra ett program i Python. Fördelen mot MATLAB är att Python effektivt utnyttjar moderna datorers förmågor att använda flera trådar parallellt. Med andra ord körs flera processer i programmet samtidigt utan kravet att tidigare operationer genomförts fullständigt. Detta i kombination med mjukvarupaket för matematiska operationer, exempelvis Numpy eller Scipy [5], gjorde Python till ett lämpligt verktyg för projektets mål.

Det visade sig dock att Python och MATLAB fungerar annorlunda ur ett numeriskt perspektiv – exempelvis gav ett ljud maximala värdet 0,25 i MATLAB och $1{,}7 \cdot 10^{-38}$ i Python. Resultatet av Python-programmets matematiska operationer var således inte alls som förväntat. Ljudet blev ohörtbart på grund av ljudfenomen som distortion och brus. Gruppen valde att ej gräva djupare i bakomliggande orsaker på grund av tidsbrist, utan istället fokusera på att få fram ett bra resultat med förinspelat ljud. Arbetet fortsatte därför i MATLAB.

MATLAB fungerar väl för redan inspelade ljud, men fungerar sämre för in- och uppspelning i realtid. Funktionen recordblocking [6] används för att spela in ljud kontinuerligt, vilket leder till att mikrofonen ständigt stängs av och på. Detta resulterar i tydliga och störande klippljud. Programmet var dessutom långsamt. För att kunna presentera ett resultat med rimlig ljudkvalitet valdes därför att inte förändra ljud i realtid. Om projektet getts mer tid skulle eventuella alternativ till recordblocking studerats.

I verklig tillämpning av en vocal harmonizer måste tonarten musikstycket spelas i vara känt för att veta när liten eller stor ters skall användas.

4.1 Felkällor

Under projektets gång har arbetet drabbats av ett par oväntade problem. Datorers oförmåga att hantera mycket små numeriska värden tvingade arbetet bort från Python. MATLAB har även stundvis svårt att vara exakt när

det kommer till numeriska beräkningar. Det finns alltså en risk att verktygen som använts påverkar resultatet.

Vidare finns det faktorer kring in- och uppspelning som påverkar hur bra den slutgiltiga skalan faktiskt upplevs. Både samplingsfrekvens och mikrofon spelar stor roll vid inspelningen, och kvalitén på hörlurar eller högtalare påverkar det upplevda resultatet.

Referenser

[1] Zölzer, U., *Digital Audio Effects*, John Wiley & Sons Ltd, 2011

[2] Schmidt-Jones, C., *Musical Intervals, Frequency, and Ratio*, Rice University, http://goo.gl/nhAhCQ, hämtad 141007

[3] Sandström, P., *Föreläsning 6 TFYA65*, Linköpings Universitet, 2014

[4] Moinet, A. & Dutoit, T., *PVSOLA: a phase vocoder with synchronized overlap-add*, http://goo.gl/N36oye, hämtad 141010

[5] *Dokumentation av Numpy och Scipy*, http://docs.scipy.org/doc, hämtad 141010

[6] *Dokumentation recordblocking, MATLAB*, http://goo.gl/GW7gUl, hämtad 141010