音频转乐谱

自 55 张蔚桐 2015011493 zwt15@mails.tsinghua.edu.cn

信号与系统小论文 2016.6.24
王旭康 2014011570
张蔚桐 2015011493——数字匹配滤波器的设计和应用

给出一段钢琴（或其他乐器）的**录音**，通过频域分析给出**乐谱**。

前段工作总结

本研究是“音频转乐谱”研究的后半部分，主要由张蔚桐完成。

在前段工作中，我们试图通过频谱分析的方法来提取每一个音的乐谱，但是受到前文提到的因素的影响，频谱的高分辨率（要求采样时间相对长）和防止多个音的重叠（要求采样时间相对短）之间存在着冲突，为排解这个冲突，我们约定了最快的音为八分音符，并通过这种截取使得在每一帧中存在的所有音都完整的存在在整个帧中，来解决不能确定音在截取帧中的具体出现位置的问题。但尽管如此，频谱分辨率仍在 2Hz 左右，对频率相差不大的低音仍存在着较大的差异，必须寻找另一解决方法。

考虑钢琴产生的机制，发现实际上每一个按键所产生的频谱是独立的，也就是说对于每一帧的音乐，其必然由在整个钢琴键盘中的 88 个音中的某几个音组成，而不会有其他的音。因此，可以通过检测 88（或可以通过一些限制（比如已知音高范围）来进一步缩小范围）个音是否存在在帧中即可。

理论分析

数字匹配滤波器

信号与系统知识告诉我们，在连续系统中，可以使用“匹配滤波器”来解决这个问题。匹配滤波器是检测目标信号是否在输入信号中存在的一种滤波器，其冲击响应为 h(t)=s(T-t),其中 s 为待检测的目标信号，滤波器要求输入信号和目标信号是有限时间的。T 取信号时域长度。

考察离散时间的情况，做一个简单的变化可以变为 h[n]=s[N-n],因此可以得到在离散时间下的冲击响应（当然这种方法实际上将连续的时域有限的输入信号和目标信号都延拓成周期的，但是后文发现效果还是很好的），通过 DFT 可以得到输出信号

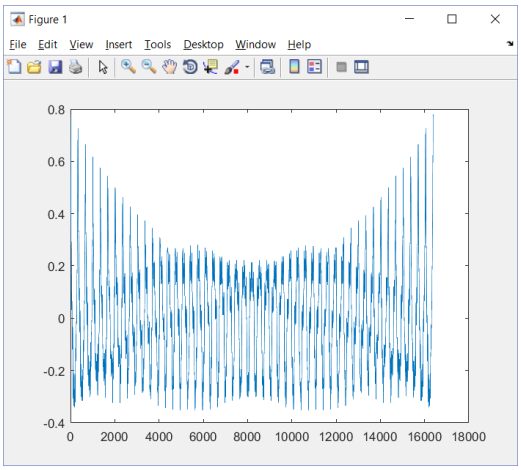
我们设计的“数字匹配滤波器” h[n]=s[N-n],以上的减法因为在 DFT 性质上应理解为模余运算，可以计算得到冲击响应的 DFT 是 H[k]=S[-k]*EXP(2*π*j*k/N)（课下作业已经证明），同样，这个-k 也理解为对 N 的模余计算。于是我们可以写出“数字匹配滤波器”的函数（MATLAB 实现，如图一）

时间复杂度分析：包括三次计算 FFT（IFFT）的过程和两次线性的向量处理工作，整体上是 N*log（N）的时间复杂度。N 是一帧的长度，可以通过一些方法使得目标音的长度和一帧的长度相等

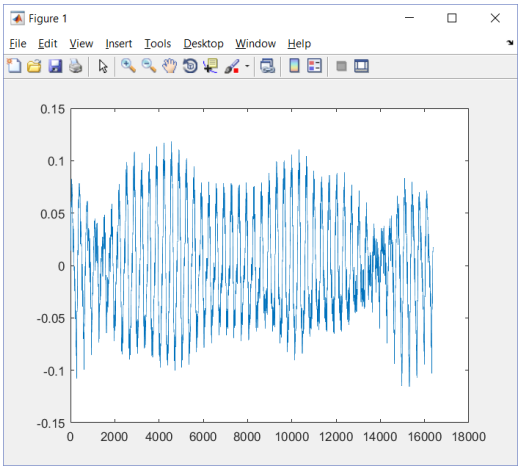
效果分析：使用了 do 作为标准音，和 do，ri，mi 三个音分别进行了数字匹配，效果如下图二到四所示,同时根据和音的匹配，效果也还可以。在没有谐波影响的情况下可以达到 10 倍的区分度，在谐波影响下可以通过设计的函数得到一部分的区分。

```
function [ X ] = match( Y,A )
B=A;
N=length(Y);
for i=N/1:1:N
    B(i)=0;
end;
% A（就是B）为目标信号
% Y是输入信号
yf=fft(Y);%输入信号的FFT
sf=fft(B);%目标信号的FFT
f=zeros(N,1);
%下面是通过目标信号的FFT选取冲击响应的FFT
f(1)=sf(1);
for j=1:1:N-1
    f(j+1)=sf(N-j+1)*exp(2*pi*1i*j/N);
end
%两者相乘得到输出信号的FFT
xf=zeros(N,1);
for i=1:1:N
    xf(i)=f(i)*yf(i);
end
%反变化得带输出信号
X=real(ifft(xf));
end
```

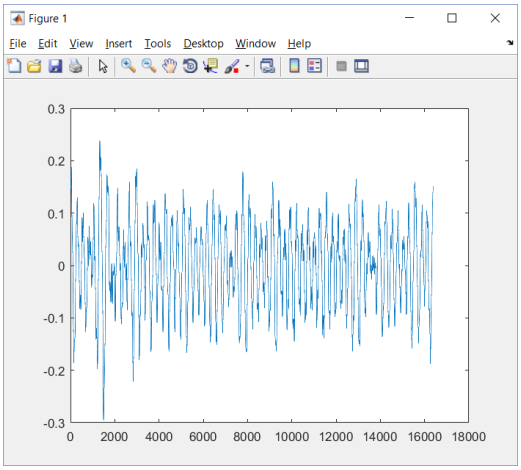
图一，数字匹配滤波器的 MATLAB 代码



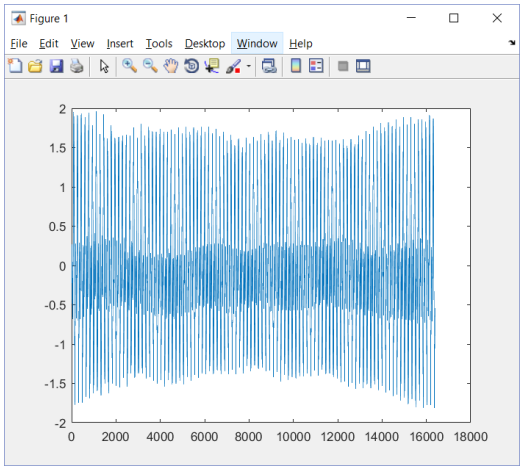
图二，do 和 do 的匹配波形，峰值可达 0.7 以上



图三，re 和 do 的匹配波形，峰值可达 0.1 以下



图三，mi 和 do 的匹配波形，峰值可达 0.3 以下



图五，八度谐波对数字匹配滤波器的影响

八度谐波对匹配滤波器的影响

如图五，是高音 do 和 do 的匹配结果，发现出现不同现象，首先，峰值本身较大，同时，发现没有像 do-do 匹配（图 2）中的中间有明显下降的问题，因此

可以认为这种谐波的匹配导致方差相对相等匹配较小，可以通过一些方差的调整来解决问题

其他误差调整

首先是因为得到的标准音存在着能量上的差异，而不同的帧之间也存在着能量的差异，因此应当通过一些方式实现归一化，一个简单的方法是考察向量夹角的方式进行归一化

Cos(α)=(a ,b)*(a ,b)/(||a||*||b||),可以通过匹配项的平方除以检测帧和输入帧自匹配的平方（在连续时间中为函数的能量谱）进行归一化。经验证，得到的效果良好。

同时因为音乐中很少出现正好差 8 度的（出现谐波）的和声，可以只在相差八度的两个匹配之间只选择一个，进一步消除谐波的影响。

编码和调试

MATLAB 的编码和调试

以下为 MATLAB 中的源码，作用是读取音乐，输出分析后的音高（以输入的标准音高为绝对音高）到文件中。

main

```
clc;
clear all;
disp('hi');
N=16384;
M=32;
[step_origin,fs]=audioread('C:\Users\Zero
Weight\Documents\GitHub\Signals-
System\music\step.wav');
F=audioread('C:\Users\Zero
Weight\Documents\GitHub\Signals-
System\music\F#.wav');
left_origin=step_origin(:,1);
left_step=left_origin(1:M*N,1);
left_mat=reshape(left_step,N,M);
right_origin=step_origin(:,1);
right_step=right_origin(1:M*N,1);
right_mat=reshape(right_step,N,M);
standard=zeros(N,M/2);
next=[8,16;9,0;10,0;11,0;13,0;14,0;15,0;1,16;2,0;3,
0;4,0;0,0;5,0;6,0;7,0;1,8];
for i=1:1:11
standard(:,i)=average(right_mat(:,i*2),right_mat(:,
i*2-1),left_mat(:,i*2),left_mat(:,i*2-1));
end
standard(:,12)=average(F(1:N,1),F(1:N,2),F(N+1:2*N,
1),F(N+1:2*N,1));
for(i=12:1:M/2-1)
standard(:,i+1)=average(right_mat(:,i*2),right_mat(
:,i*2-1),left_mat(:,i*2),left_mat(:,i*2-1));
end
[music_origin,fs]=audioread('C:\Users\Zero
Weight\Documents\GitHub\Signals-
System\music\school_song_double_C.wav');
left_music_origin=music_origin(:,1);
n=floor(length(left_music_origin)/N);
left_music_step=left_music_origin(1:n*N,1);
left_music_mat=reshape(left_music_step,N,n);
right_music_origin=music_origin(:,1);
right_music_step=right_music_origin(1:n*N,1);
right_music_mat=reshape(right_music_step,N,n);
music=zeros(N,n);
for i=1:1:n
music(:,i)=combine(right_music_mat(:,i),left_music_
mat(:,i));
end
mat=zeros(16,n);
for I=1:1:n
    for J=1:1:16

mat(J,I)=ave(match(standard(:,J),music(:,I)))*ave(m
atch(standard(:,J),music(:,I)))/ave(match(music(:,I
),music(:,I)))/ave(match(standard(:,J),standard(:,J
)));
    end
end
answer=zeros(8,n);
div=0.2;
for I=1:1:n
    all=1;
    max=0;
    for J=1:1:16
        if max<mat(J,I)
            max=mat(J,I);
        end
    end
    while true
        rank=1;
        for J=1:1:16
            if mat(J,I)>mat(rank,I)
                rank=J;
            end
        end
        if mat(rank,I)<div*max
            break;
        end
        answer(all,I)=rank;
        all=all+1;
        mat(rank,I)=0;
        for K=1:1:2
            if next(rank,K)>0
                mat(next(rank,K),I)=0;
            end
        end
    end
end
txt='.\output.txt';
f=fopen(txt,'w');
for i=1:1:n
    for j=1:1:8
        fprintf(f,'%d ',answer(j,i));
    end
     fprintf(f,'\n');
end
fclose(f);
```

均值 ave（处理谐波影响）

```
function [ x ] = ave( A )
%UNTITLED5 Summary of this function goes here
%   Detailed explanation goes here
x=0;
for i=1:1:length(A)
    x=x+A(i);
end
x=x/length(A);
y=0;
for i=1:1:length(A)
    y=y+A(i)*A(i);
end
y=y/length(A);
x=y-x*x;
% x=A(length(A));
End
```

复合求均值 combine（）

```
function [ X ] = combine( A,B )
%UNTITLED3 Summary of this function goes here
%   Detailed explanation goes here
fa=fft(A);
fb=fft(B);
fx=fa+fb;
X=ifft(fx);
end
```

输出（部分）

```
8 3 5 0 0 0 0 0
3 0 0 0 0 0 0 0
8 3 0 0 0 0 0 0
10 0 0 0 0 0 0 0
13 1 0 0 0 0 0 0
3 13 0 0 0 0 0 0
```

C# 的 GUI 界面

得到处理的答案后使用 C#做了一个 GUI 界面，具体的实现方式和本课程无关，代码从略，实现的效果请见视频文件

误差分析和整理

已知的避免误差的方法

首先是读入数据的时候因为某些前段的原因导致数据可能在频谱上存在着一定的噪声，因此对于标准音阶使用了两帧作为一个音阶，利用左右声道的 DFT 加和来一是实现音量的增强，二是相对弱化一些随机的噪声。这些功能在上面个的 combine 函数中得到了解决。

其次是谐波高频的影响，使用了方差作为另外一种判据，同时使用了 “不同时存在两个相差八度的音” 的方法，这种方法的处理相对比较粗糙，但是结果还是可以接受的。

其他的未解决的误差

首先是存在着音的延时问题没有得到解决，有些较强的音在延时之后会影响下一帧的判断。考虑使用一个延时模型进行逆向的消除（通过已知前一帧的音频消去这一帧中存在的延时音），但是对于不同力度的音来说，延时的时间不同，另外演奏者可以在演奏时通过一些方式消除延时，这样使得延时系统很难设计

其次是存在着低音成分太强使得程序认为高音部分是噪声的情况，这种问题考虑的解决方法是通过适当预处理增强高音部分来解决，但是同样存在一些高音比较强的问题，必能一概而论，而是根据当时的演奏者的音高的力度决定的。因此较难解决。

整体的实验精度在 70%以上，对没有和弦的单音精度基本可以达到 90%以上。

匹配滤波器的原理的理解

可以认为匹配滤波器是将输入信号和待测信号进行一种积分（离散则为求和），由于三角函数的正交性，单纯的三角函数之间做匹配一旦不相同一定为 0，但是因为前文提到钢琴的每一个音存在着衰减较慢的谐波分量，不同的音之间可能存在着相同或较近的谐波分量，因此匹配之后并不是零，尤其是相差八度的音之间仅仅相差一个低音的基频，因此相差不大，但是基频 的意

义在于其相对与高频的变化速度较慢，同时分量较大，容易引起输出波形较大的方差因此，上文的 ave()函数处理基频是合理的。

总结

通过对钢琴的和弦的研究，实现了从频谱检测（主要由王旭康学长完成）和数字匹配滤波器（主要由本人完成）的两种检测技术的实现，同时完成了一个基于 C#的简单界面的设计（主要有本人完成）。从效果上来来看，数字匹配滤波器的效果较好，但是数字匹配滤波器的速度也较慢（实验发现），了解了音乐的频谱特性，同时自己完成了数字匹配滤波器的设计，收获很大。