基于数字滤波的音乐均衡器设计

18 数科 叶家辉 201800830004

一、摘要

在演出进行中，或者日常处理音频信号时，可能需要平衡不同频率的声音成分以达到最佳听觉效果，本课题基于数字滤波的相关知识设计了一款 mp3 音乐均衡器，可以对音乐信号中不同频带的声音进行保留或滤除，如想要听乐曲中弦乐的演奏效果则通过拖动滑动条把低频信号滤除，反之如果想听打击乐的演奏效果则滤除高频信号。这项技术可供音乐调音处理使用，也可供交响乐团或合唱团在训练中使用。本设计把人耳听觉范围（20Hz～20kHz）分成九个区间，使用者可以根据需要，自由调节九个区间内音频信号的滤除程度，对比滤波前后的波形图，并且播放滤波前后的音乐信号。然而本设计目前还存在一定缺陷，一旦开始播放一段音乐信号，就必须等待当前音乐播放完成，否则多段声音会混在一起，因此在实验时建议导入的声音信号时长尽量不要超过 15 秒。

二、引言

对于已经发行的歌曲，音乐制作人会将不同频率的声音均衡到最佳状态。但是对于一些低端耳机而言，可能没有办法把这种微妙的组合发挥到极致。比如知乎网友“mikiya”提到，他使用某耳机听歌时，“听歌全是低频混成一团，贝斯和底鼓也全都糊在一起了，人声几乎等于没有”，而经过均衡调整后，“一个原本没法听的 49 块钱耳机，至少要比 100 价位的还要好听”。可见，音乐均衡器可以结合听者的喜好，在很大程度上发挥耳机的机能。另一方面，如果我们想要发布自己的音乐作品，音乐均衡也是尤为重要的。比如根据作品风格（流行、古典、摇滚、蓝调等），对不同频段声音作出调整。

本项目所选取的九个滤波频段的音感特征：30～60Hz—沉闷；60～100Hz—沉重；100～200Hz—丰满；200～500Hz—力度；500～1kHz—明朗；1k～2kHz—透亮；2k～4kHz—尖锐；4k～8kHz—清脆；8k～16kHz—纤细。

该设计的核心原理已经被实现过（许多音乐软件就自带均衡器），本项目是基于类似原理借助 Matlab 的一次复现，并在一些方面作出了创新。

三、方案与设计思路

（一）背景介绍

该项目的具体实现一般来说有三类方法：

1. FFT 滤波。首先将音乐信号做傅立叶变换得到信号频谱，然后根据均衡器的滑动条长度对频域滤波器的值进行定义，再镜像翻转得到虚频上的定义，最后将信号频谱与频域滤波相乘做傅立叶逆变换。

2. 滤波器组。让九个频带每个对应一个带通滤波器，滤波完成后的输出值再乘以一个系数，该系数与滑动条的长度有关。

3. FIR2。按照滑动条长度得到每个点归一化频率的幅度值，可以直接设计出该状态下的滤波器，再用这个滤波器对信号进行滤波。

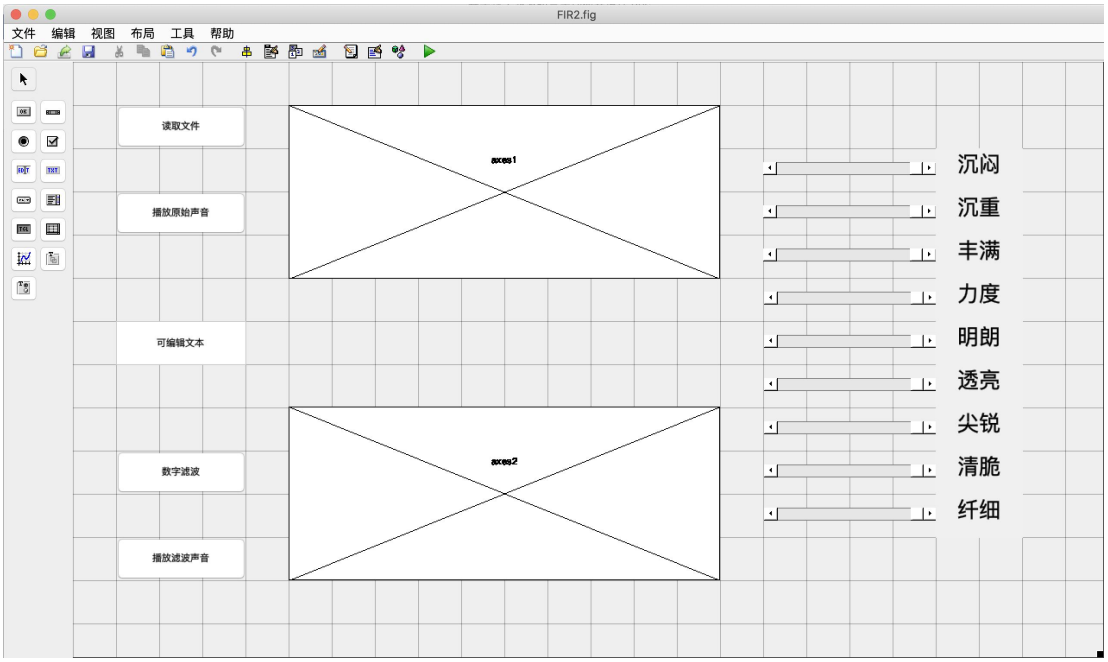
由于 FIR2 方法在实践中较为简单和有效，因此本项目即采用这种方法。

（二）实现过程

1. 新建 Matlab 的 GUI 编程窗口，添加需要的控件。

包括四个普通按钮（分别用于读取文件、播放原始声音、数字滤波、播放滤波声音），一个可编辑文本（用于显示当前读取的文件名），两个图标区（分别用于显示原始声音波形和滤波声音波形），九个滑动条（分别对应九种频段声音的滤除程度）和九个静态文本（分别对应九个滑动条，用以提示该滑动条调节的频率区间的听感）。

添加完毕后，将所有控件按照下图更改名称，方便用户辨识。



2. 为“读取文件”按钮添加回调函数。

代码的具体含义已写在注释中。用户选择文件后，会将文件名显示在可编辑文本中，读入的声音信号波形会显示在上面的坐标图中。

```
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global abc;  % 文件名
global Fs;   % 采样频率
global x;    % 数据

% 选择mp3文件
[FileName,PathName] = uigetfile({'*.mp3'},'File Selector');
set(handles.edit1,'String',FileName);  % 把可编辑文本的值设置为文件名
abc = fullfile(PathName,FileName);

% 读取数据
[x,Fs] = audioread(abc);
dt = 1.0/Fs;
N = length(x);
T = N * dt;
t = linspace(0,T,N);
% 把读取到的数据画在第一个表格图中
plot(handles.axes1,t,x);
```

3. 为“播放原始声音”按钮添加回调函数。

这一步比较简单，代码的具体含义已写在注释中。用户点击该按钮后，程序会把刚才读入的数据播放出来。

```
% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global x; global Fs;
sound(x,Fs);
```

4. 为“数字滤波”按钮添加回调函数。

这一步是整个均衡器的核心。

(1) 用 9 个变量分别读取九个滑动条的数值。

(2) 生成滤波器定义数组。f 表示每个滑动条对应的频段，m 数组储存滑动条的当前长度。

(3) 将这两个数组代入 fir2 函数即可设计多通带任意响应的 FIR 滤波器。

(4) 通过 filter 函数对原数据按照设计的滤波器进行滤波。

(5) 把滤波完成的信号画在第二个坐标区域。

```
% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global Fs;
global x;
global y;
% 读取数字均衡器设定
v1 = get(handles.slider1,'Value');
v2 = get(handles.slider2,'Value');
v3 = get(handles.slider3,'Value');
v4 = get(handles.slider4,'Value');
v5 = get(handles.slider5,'Value');
v6 = get(handles.slider6,'Value');
v7 = get(handles.slider7,'Value');
v8 = get(handles.slider8,'Value');
v9 = get(handles.slider9,'Value');
f = [0,0.0028,0.0057,0.0113,0.0227,0.0454,0.0907,0.1814,0.3628,0.7256,1];
m = [0,v1,v2,v3,v4,v5,v6,v7,v8,v9,0];
b = fir2(100,f,m);
y = filter(b,1,x);
N = length(x);
dt = 1.0/Fs;
T = N*dt;
t = linspace(0,T,N);
plot(handles.axes2,t,y);
```

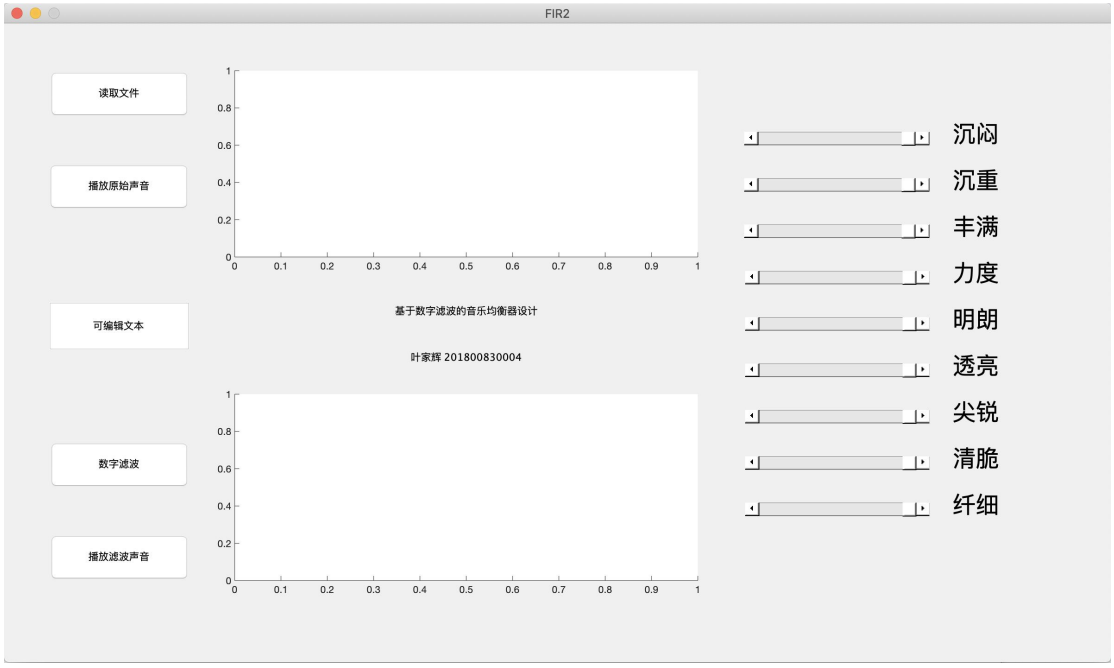
5. 为“播放滤波声音”按钮添加回调函数。

这一步比较简单，代码的具体含义已写在注释中。用户点击该按钮后，程序会把刚才滤波完成的数据播放出来。

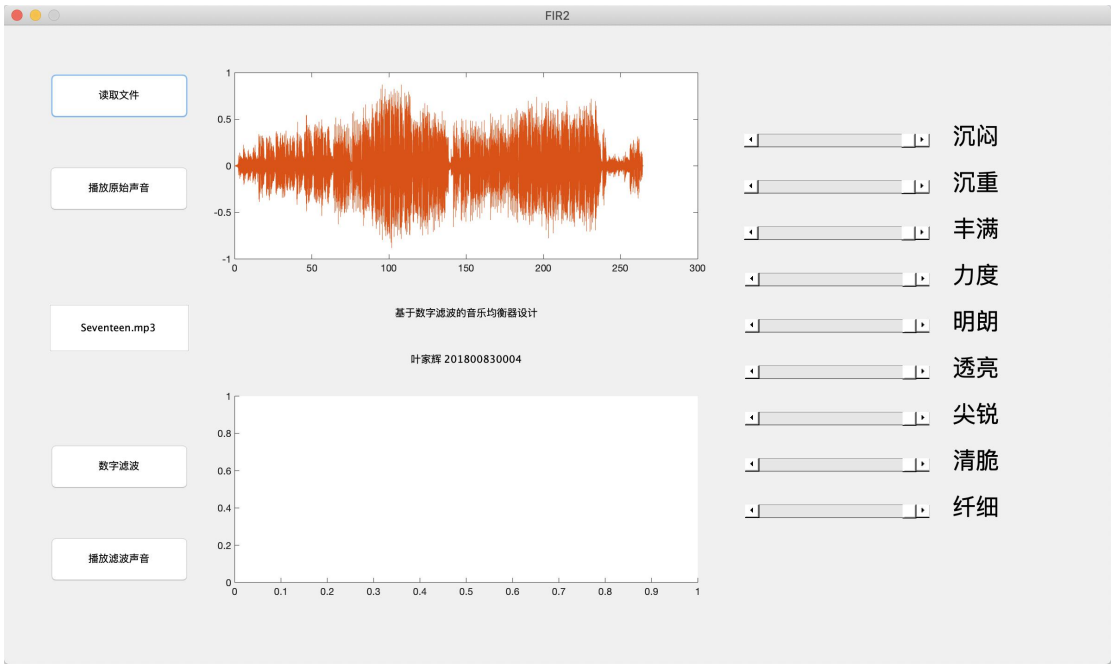
```
% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global y; global Fs;
sound(y,Fs);
```

四、设计结果

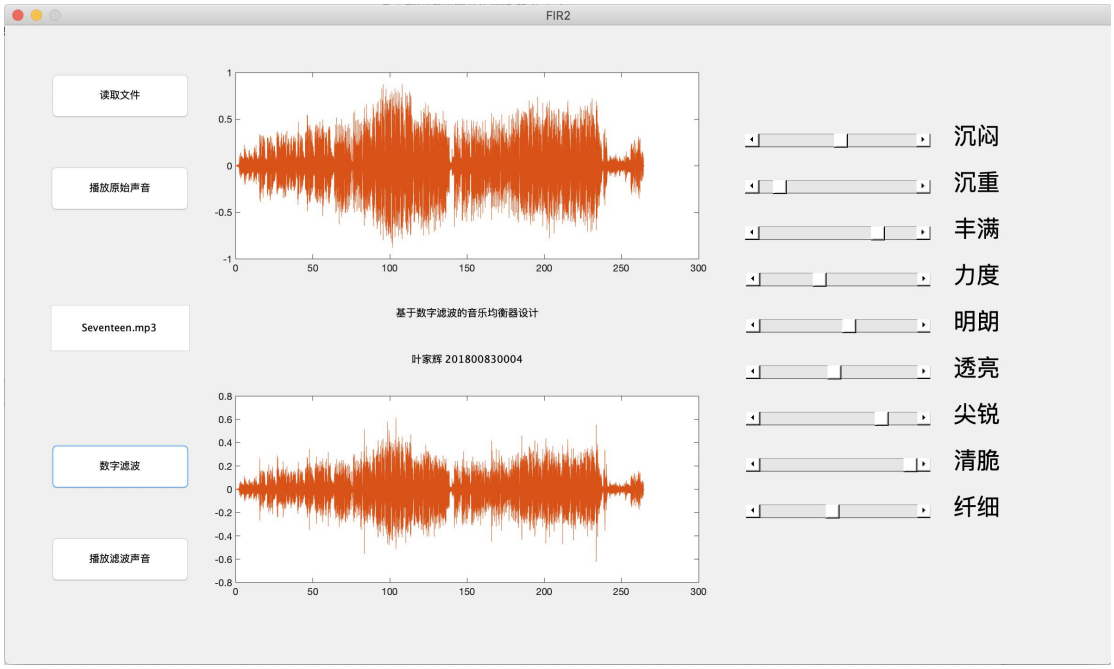
初始页面



读取音乐文件



拖动滑块根据需要对信号进行均衡



点击“播放原始声音”和“播放滤波声音”即可播放滤波前后的音乐

五、总结

从上面的展示中可以看到本项目取得了预期的效果，并且经多次实验证明该结果是可重复的。

本项目综合了数字滤波与滤波器设计的相关知识，并结合 Matlab 程序设计对核心原理进行实现，借助 Matlab 图形化编程工具对原理实现的全过程进行了可视化展现。

本项目的核心算法就是根据 FIR2 方法进行滤波器设计，即根据用户拖动的滑动条长度设定得到归一化频率下的每一点的幅度值，设计出一个能够满足这种设定的滤波器，即可用该滤波器对信号滤波后进行输出。这种方法相对较为简单，在实际应用中也取得了不错的效果。但是实际应用中的数字滤波器与理想滤波器毕竟存在一些差距，所以难以达到 100%的滤波效果。

本项目所有自己设计的代码（不含 Matlab 图形化编程自动生成的无关代码）均已呈现在附录中。其中参考网上的部分已在注释中标注。

附录：

软件运行环境：macOS 10.15.4，Matlab R2019B。

源代码：

1. “读取文件”按钮回调函数

```
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global abc;  % 文件名
global Fs;   % 采样频率
global x;    % 数据

% 选择 mp3 文件
[FileName,PathName] = uigetfile({'*.mp3'},'File Selector');
set(handles.edit1,'String',FileName);  % 把可编辑文本的值设置为文件名
abc = fullfile(PathName,FileName);

% 读取数据
[x,Fs] = audioread(abc);
dt = 1.0/Fs;
N = length(x);
T = N * dt;
t = linspace(0,T,N);
% 把读取到的数据画在第一个表格图中
plot(handles.axes1,t,x);
```

2. “播放原始声音”按钮回调函数

```
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global x; global Fs;
sound(x,Fs);
```

3. “数字滤波”按钮回调函数

```
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global Fs;
global x;
global y;
% 读取数字均衡器设定
v1 = get(handles.slider1,'Value');
v2 = get(handles.slider2,'Value');
v3 = get(handles.slider3,'Value');
v4 = get(handles.slider4,'Value');
v5 = get(handles.slider5,'Value');
v6 = get(handles.slider6,'Value');
v7 = get(handles.slider7,'Value');
```

```
v8 = get(handles.slider8,'Value');
v9 = get(handles.slider9,'Value');
% f 表示每个滑动条对应的频段，m 储存每个滑动条当前的长度
f = [0,0.0028,0.0057,0.0113,0.0227,0.0454,0.0907,0.1814,0.3628,0.7256,1];
m = [0,v1,v2,v3,v4,v5,v6,v7,v8,v9,0];
%  根据频率向量和对应频率点上的幅度定义滤波器
%  这一步设计滤波器的时候 fir2 函数的写法参考了网上
b = fir2(100,f,m);
%  再根据定义的滤波器进行滤波
y = filter(b,1,x);
N = length(x);
dt = 1.0/Fs;
T = N*dt;
t = linspace(0,T,N);
plot(handles.axes2,t,y);
```

4. “播放滤波声音”按钮回调函数

```
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global y; global Fs;
sound(y,Fs);
```