声之趣——一款音频处理小工具

颜磊　怡喆扬　缪晨轩　黄翔宇

指导老师：赵均

摘要—这是《信号分析与处理》课程期末大作业报告。我们小组通过运用高通滤波器、低通滤波器、切比雪夫滤波器等课内所学知识，实现了一个简易的音频均衡器；此外，我们通过自身探索和调研，使用了梅尔频率特征提取、动态时间规整等课内没有过多涉及的知识，自主设计完成了简易的“听歌识曲”小任务，同时也参考了很多资料，自学了 Matlab GUI 的制作，最终成功将所实现的函数与 GUI 结合，制作出了一款音频处理小工具。

关键词—信号分析与处理; 特征提取; 特征匹配; 滤波器;Matlab GUI

Abstract—This is the final assignment report for signal analysis and processing. Our group has developed a simple audio equalizer using high-pass filter, low-pass filter, Chebyshev filter, and other knowledge learned in class, using Mel frequency feature extraction, Dynamic time warping, and other knowledge not too much involved in the class, designed to complete the simple "Listening to songs to recognize the songs" small tasks, but also referred to a lot of information, self-learning the production of MATLAB GUI, the final successful function and the GUI to achieve the combination of an audio processing tool.

Keywords—Signal analysis and processing; feature extraction; feature matching; filter; MATLAB GUI

Contents

1. 均衡器设计

1.1. 滤波器简介

滤波器是由各种电子元器件组成的滤波电路。滤波器可以对电源线中特定频率的频点或该频点以外的频率进行有效滤除，得到一个特定频率的电源信号，或消除一个特定频率后的电源信号。滤波器按照所处理的信号可以分为模拟滤波器和数字滤波器两种，按照所处理的信号则可分为低通、高通、带通、带阻等类型。

我们的目标是对原始音频信号实现均衡器的效果。因此，我们首先可以利用滤波器滤出音频信号不同频率的分量，之后，对这些分量进行加权叠加，即可得到音调均衡后的信号。

下面就是选择滤波器的种类，由于 matlab 通过采样得到音频信号，因此我们需要选择数字滤波器。数字滤波器分为 FIR 滤波器（有限脉冲响应）和 IIR 滤波器（无限脉冲响应）。

使用 FIR 滤波器可以最大程度保持音质，因为 FIR 滤波器的相位响应是线性的。但是 FIR 滤波器的计算量很大，主要是和音频信号的采样率以及均衡器的分辨率有关。而 IIR 滤波器就在于达到相同的性能下，所需的阶数更小，但是它计算出的信号相位是非线形的，不同的频率的信号相位偏差不一致，导致最后叠加出的信号也有很大的变化，因此它对音质有一些损伤。在这里，由于我们只实现简易均衡器，主要目的是对音频信号进行适度的改变，测试听歌识曲功能的好坏，因此我们不考虑 IIR 滤波器损失的小部分音质，分别设计低通、带通、高通 IIR 滤波器（切比雪夫 II 型）。

三个滤波器频率指标如下：

- 低通：0-200Hz
- 带通：200-4000Hz
- 高通：4000Hz 以上

需要注意的是，在实际设计滤波器时，三者之间可能会出现交汇，为了保证最后叠加时交汇处既不会塌陷也不会冒出，因此在设计时我们需要通过多次实验修改滤波器参数。

1.2. 代码实现

```matlab
% IIR低通滤波器
function y1 = lp(x, fs)
    maxFreq = fs / 2;  % 计算最大频率

    % 设计低通滤波器参数
    [lp_N, lp_Wn] = cheb2ord(180 / maxFreq, 360 / maxFreq, 1, 40);
    [lp_b, lp_a] = cheby2(lp_N, 40, lp_Wn, 'low');

    % 以下实现绘制滤波器特性
    % figure
    % freqz(lp_b, lp_a);
    % title("lowpass");

    % 应用滤波器
    y1 = filter(lp_b, lp_a, x);
end
```

Code 1. 低通滤波器

```matlab
% IIR带通滤波器
function y2 = bp(x, fs)
    maxFreq = fs / 2;  % 计算最大频率

    % 设计带通滤波器参数
    [bp_N, bp_Wn] = cheb2ord([400, 3000] / maxFreq, [150, 5000] / maxFreq, 1, 40);
```

```
    [bp_b, bp_a] = cheby2(bp_N, 40, bp_Wn, '
    bandpass');

    % 以下实现绘制滤波器特性
    % figure
    % freqz(bp_b, bp_a);
    % title("bandpass");

    % 应用滤波器
    y2 = filter(bp_b, bp_a, x);
end
```

Code 2. 带通滤波器

```
% IIR高通滤波器
function y3 = hp(x, fs)
    maxFreq = fs / 2;  % 计算最大频率

    % 设计高通滤波器参数
    [hp_N, hp_Wn] = cheb2ord(4000 / maxFreq, 2500 /
     maxFreq, 1, 40);
    [hp_b, hp_a] = cheby2(hp_N, 40, hp_Wn, 'high');

    % 以下实现绘制滤波器特性
    % figure
    % freqz(hp_b, hp_a);
    % title("highpass");

    % 应用滤波器
    y3 = filter(hp_b, hp_a, x);
end
```

Code 3. 高通滤波器

至于实现均衡器效果，我们将得到的低频、中频、高频信号进行加权叠加即可，例如低音增强时对低频信号乘以三倍，这里也需要不断进行调试以选择最好的效果。除此之外，我们增加信号的快慢放及倒放来增加均衡器的丰富度，分别通过更改采样率和倒转信号矩阵得到。代码如下：

```
% 该函数实现对采集后的音频信号进行快慢放处理
function velocity(x, fs, N)
M=N*fs
sound(x,M)
end
% 该函数实现信号倒放
function sideDown(x,fs)
y0=flipud(x);
sound(x,fs);
end
```

Code 4. 调音快慢放

2. 特征提取

本次作业中通过 mfcc（梅尔频率倒谱系数）的提取作为特征提取的方法。人的耳朵可以看成是一组滤波器，耳朵的滤波作用经研究是在对数域的尺度上进行的。MFCC 特征就是依据人耳的滤波机制来设计的，以达到模拟人耳的目的。

2.1. 梅尔频率简介

频率的单位是 Hz，人耳能听到的频率范围是 20-20000Hz，但人耳对 Hz 这种标度单位并不是线性感知关系。例如如果我们适应了 1000Hz 的音调，如果把音调频率提高到 2000Hz，我们的耳朵只能觉察到频率提高了一点点，根本察觉不到频率提高了一倍。也就是说人耳对低频音调的感知较灵敏，在高频时人耳是很迟钝的。如果将普通的频率标度转化为梅尔频率标度，映射关系如下图所示：

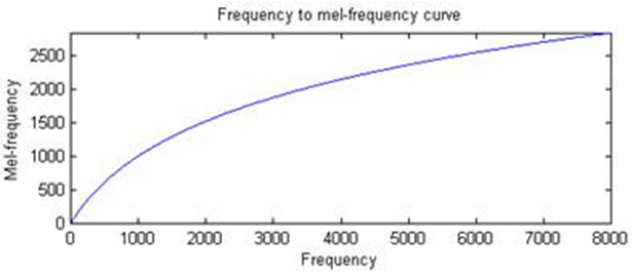


Figure 1. 梅尔映射关系

$$mel\left(f\right)=2595\times\log_{10}\left(1+\frac{f}{700}\right)$$

在梅尔标度下，如果两段语音的梅尔频率相差两倍，则人耳可以感知到的音调大概也相差两倍。

2.2. 倒谱

倒谱图是用来“提取”语音的音色（timbre）的，音色是区分说话人最有力的特征。下面给出求倒谱的公式：

$$C\left[x\left(n\right)\right]=F^{-1}\left[\log\left(\left|F\left(x\left(n\right)\right)\right|^{2}\right)\right]$$

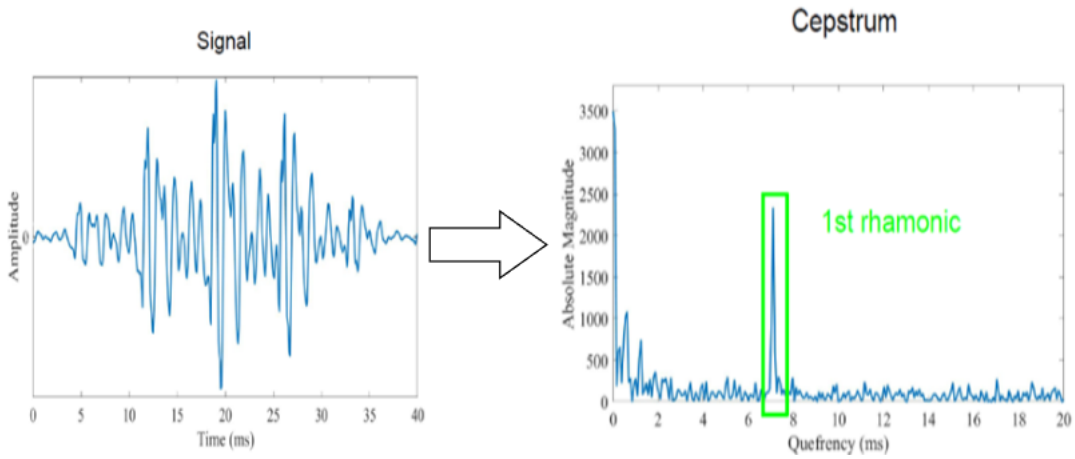


Figure 2. 逆变换所得倒谱图

最后逆变换得到的倒谱图，横坐标为倒频率，纵坐标是振幅。1st harmonic，是从倒谱图的右边往左看的第一个尖峰，实际上，这个 1st harmonic 对应原始信号的基频。之所以倒谱图能提取音色，是利用了离散傅里叶变换 (DCT) 和时域卷积的性质。人的发声过程可以看作是肺里的气流通过声带这个线性系统。如果用 $e(t)$ 表示输入声音的音高，$h(t)$ 表示声带的响应（也即我们需要获取的语音特征），那么听到的语音信号 $x(t)$ 即为二者的卷积：

$$x(t) = h(t) * e(t)$$

离散化之后，再进行离散傅里叶变换，时域卷积等价于频域乘积：

$$X(n) = H(n) \cdot E(n)$$

接下来，先取幅值，再取平方，最后取对数：

$$\log\left[|X(n)|^2\right] = 2\log|H(n)| + 2\log|E(n)|$$

现在已经将语音信号，分解成两个信号的和了，最后一步是是用离散傅里叶逆变换，得到倒谱图。

2.3. 特征提取

MFCC 特征就是依据人耳的滤波机制来设计的，以达到模拟人耳的目的，具体的提取过程如下图所示：

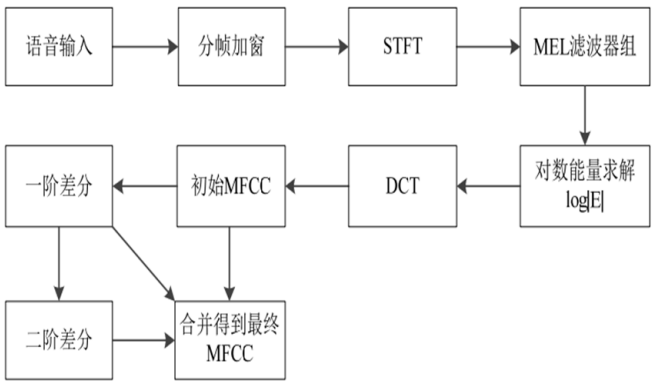


Figure 3. MFCC 特征提取流程

初始的 MFCC 系数能很好的表达音频的静态特性，而音频信号处理过程中通常也需要反映音频变化趋势的动态特性。动态特性可通过对初始 MFCC 系数进行差分运算得到。研究表明动、静态特征的结合能达到更好的识别性能。在这次大作业中，我们利用了三种办法实现 MFCC 特征的提取，第一种是自己编写的提取函数，流程如上图所示；第二是使用 voicebox 工具箱中提供的函数；第三种是 matlab 自带的函数。

```
audio_mfcc = mfcc_m(audio,fs,24,numCoeffs,
    windowLength,overlapLength);
audio_mfcc = v_melcepst(audio,fs,'E0dD',numCoeffs
    ,24,windowLength,overlapLength);
audio_mfcc = mfcc(audio, fs, "Window", win,'
    LogEnergy', 'Ignore', ...
     'NumCoeffs', numCoeffs,'OverlapLength',
    overlapLength);
```

Code 5. 三种 MFCC 提取方法

以下是自己编写的 MFCC 代码：

```
% 对输入的语音序列x进行MFCC参数的提取，
% 返回MFCC参数和一阶、二阶（可选）差分MFCC参数，Mel
    滤波器的个数为p，
% MFCC参数个数L，一般取12-16；采样频率为fs
% 按帧长为n点分为一帧，相邻两帧之间的帧移为inc
bank = v_melbankm(p,framesize,fs,0,0.5,'t');
% 归一化Mel滤波器组系数
bank = full(bank); % 将稀疏矩阵转换为满存储
bank = bank / max(bank(:));
% 计算DCT系数 L*p
for k = 1:L
    n = 0:p-1;
    dctcoef(k,:) = cos((2*n+1)*k*pi/(2*p));
end
% 归一化倒谱
w = 1 + 0.5 * L * sin(pi*[1:L]./L);
w = w / max(w);
% 预加重滤波器
xx = double(x);
xx = filter([1,-0.97],1,xx);
% 语音信号分帧
xx = v_enframe(xx,framesize,inc);
n2=fix(framesize/2)+1;
```

```
% 计算每帧的MFCC参数
for i = 1:size(xx,1)
    y = xx(i,:);
    s = y' .* hamming(framesize);
    t = abs(fft(s));
    t = t .^ 2;
    c1 = dctcoef * log(bank*t(1:n2));
    c2 = c1 .* w';
    m(i,:) = c2;
end
% 求一阶差分系数
dtm = zeros(size(m));
for i = 3:size(m,1)-2
    dtm(i,:) = -2 * m(i-2,:) - m(i-1,:) + m(i+1,:)
    + 2 * m(i+2,:);
end
dtm = dtm / 3;
% 求取二阶差分系数
dtmm = zeros(size(dtm));
for i = 3:size(dtm,1)-2
    dtmm(i,:) = -2 * dtm(i-2,:) - dtm(i-1,:) + dtm(
    i+1,:) + 2 * dtm(i+2,:);
end
dtmm = dtmm / 3;
% 合并mfcc参数和一阶、二阶差分mfcc参数
ccc = [m dtm dtmm];
% 去除首尾几帧避免出现差分为0
ccc = ccc(101:size(m,1)-100,:);
```

Code 6. 本组编写的 MFCC

3. 特征匹配

3.1. 动态时间规整简介

DTW 是一种衡量两个时序序列之间相似度的算法。它允许在时间轴上进行非线性拉伸或压缩，以便找到两个序列之间的最佳匹配。DTW 常用于语音识别、时间序列分析和手写识别等领域。DTW 通过计算两个序列之间每对点（或向量）的距离来构建累积距离矩阵，以下图为例：

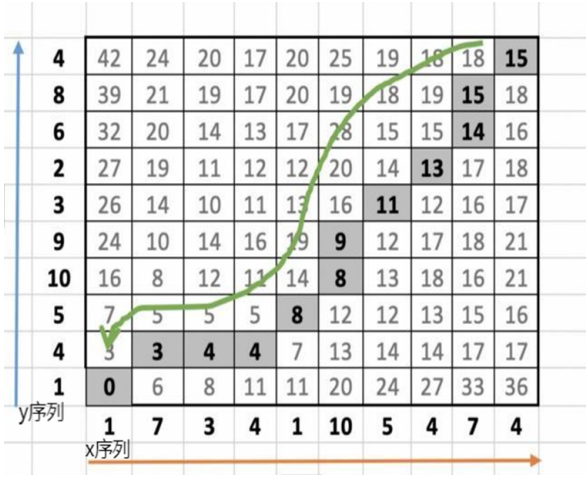


Figure 4. 累积矩阵计算

对于 x、y 两个序列，其累积距离矩阵中的每个元素，都是其对应两点之间的距离与其左下三个元素中最小值之和。计算得到的最后一个元素（右上角）即为这两个序列的 DTW 累积距离。其匹配则是从最后一个元素开始回溯，依次寻找其左下三元素中的最小值，得到的路径则代表这两个序列的最佳对齐方式，对应最小化的总距离。

对于本次作业中的 MFCC 特征而言，每一帧的 MFCC 特征都是一个 n 维向量，其距离可以通过欧氏距离或曼哈顿距离来衡量。

3.2. 代码实现

首先是距离矩阵的计算，这里使用的是欧氏距离：

```
function dtw_distance = dtw_m(matrix1, matrix2)
    % matrix1: 第一个矩阵，大小为 M x N
    % matrix2: 第二个矩阵，大小为 P x N

    % dtw_distance: DTW距离

    [M, N] = size(matrix1);
    [P, ~] = size(matrix2);

    % 计算距离矩阵
    distance_matrix = zeros(M, P);
    for i = 1:M
        for j = 1:P
            distance_matrix(i, j) = norm(matrix1(i,
     :) - matrix2(j, :));
        end
    end
```

Code 7. 使用欧式距离

也可以使用曼哈段距离：

```
% 计算曼哈顿距离矩阵
manhattan_matrix = zeros(M, P);
for i = 1:M
    for j = 1:P
        manhattan_matrix(i, j) = sum(abs(matrix1(i,
     :) - matrix2(j, :)));
    end
end
```

Code 8. 使用曼哈顿距离

然后是累积距离矩阵的计算：

```
% 初始化累积距离矩阵
    acc_distance = zeros(M, P);
    acc_distance(1, 1) = distance_matrix(1, 1);

    % 计算累积距离矩阵
    for i = 2:M
        acc_distance(i, 1) = distance_matrix(i, 1)
    + acc_distance(i-1, 1);
    end
    for j = 2:P
        acc_distance(1, j) = distance_matrix(1, j)
    + acc_distance(1, j-1);
    end
    for i = 2:M
        for j = 2:P
```

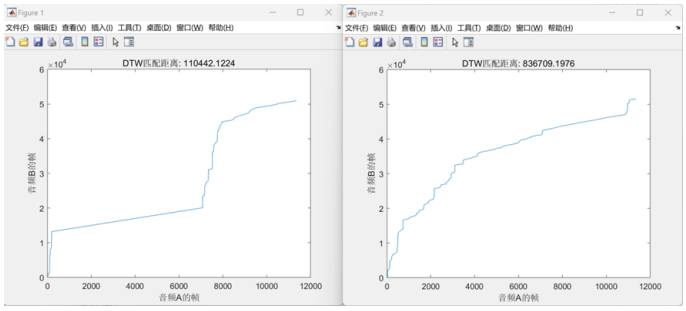
```
            acc_distance(i, j) = distance_matrix(i, j) + min([acc_distance(i-1, j), acc_distance(i, j-1), acc_distance(i-1, j-1)]);
        end
    end

    dtw_distance = acc_distance(M, P);
```

Code 9. 累积矩阵计算

最后该函数返回两个等行矩阵的 DTW 累积距离，考虑到矩阵规模对结果大小的影响，结果距离除以了两个矩阵大小之积。

3.3. 效果展示

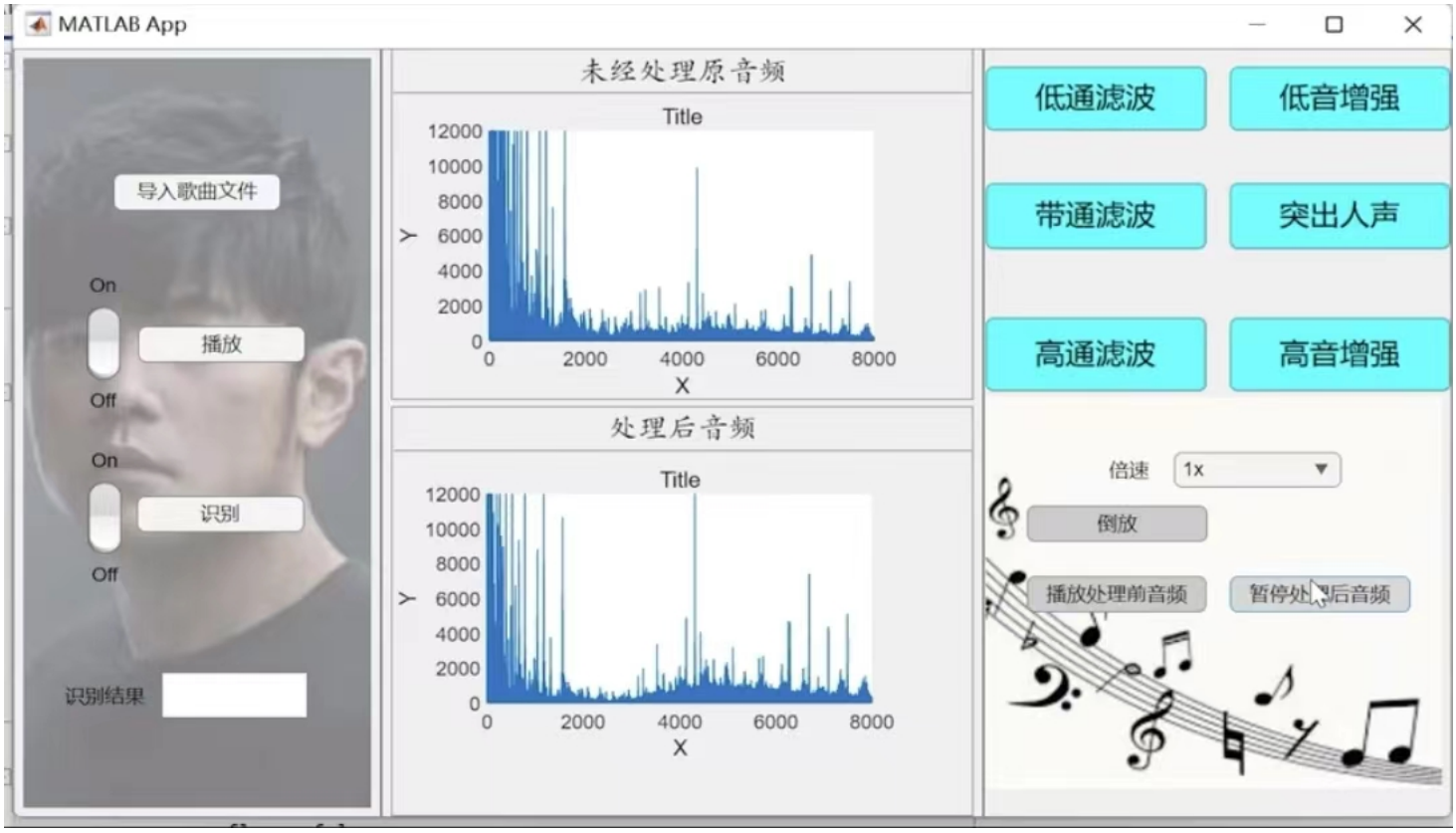
下面的图片展示了 DTW 算法对于两个序列的匹配结果以及 DTW 匹配距离的计算，其中左边为《稻香》片段与原曲的匹配，右边为该片段与《只因你太美》的匹配，可以看到左图中的直线是匹配度最高的一部分。



由于我们自己编写的函数并没有实现路径匹配的可视化功能，本图使用 Matlab 自带的 dtw 函数进行绘制。

4. GUI 制作及最终效果

我们通过利用 Matlab，制作了相应的 GUI，可以将上述实现的函数封装在 GUI 中，实现了一款音频处理小软件。



GUI 制作中用到的部分关键函数如下：

- 作图函数

```
%给出音频数组和采样率，绘制音频时域和频域图
function draw(site,x,fs)
    N=length(x);
    n=N-1;
    t=(0:N-1)/fs;
    X=fft(x);
    f=fs/N*(0:round(N/2)-1);
    plot(site,f,abs(X(1:round(N/2))));
%    axis([0 3000 0 max(f)]);
    xlabel('Frequency/(s)');ylabel('Amplitude')
    ;
    title('信号的频谱');
    grid;
end
```

Code 10. 作图函数

- 播放属性控制

```
properties (Access = public)
stop_flag = false;
pause_flag = false;
origin_flag = false; % Description
end
```

Code 11. 播放控制属性

- 按钮回调函数

```
% Button pushed function: Button
        function ButtonPushed(app, event)
            [file,path] = uigetfile('*.*');
            app.music_path = fullfile(path,file
    );
            [app.audiodata,app.fs] = audioread(
    app.music_path);
            app.audiodata_origin = app.
    audiodata;
            app.fs_origin = app.fs;
        end
```

Code 12. 按钮回调函数

5. 总结与展望

我们小组通过运用高通滤波器、低通滤波器和切比雪夫滤波器等课程中学习到的知识，成功实现了一个简易的音频均衡器。这一过程不仅巩固了我们对基础信号处理技术的理解，还提升了我们的实际应用能力。此外，我们在课外自主学习和探索中，使用梅尔频率特征提取和动态时间规整等技术，完成了简易的“听歌识曲”任务。这些知识在课程中并未详细讲授，但我们通过调研和实验，掌握了相关理论和实现方法，展示了自主学习的能力。在此过程中，我们还学习并应用了 Matlab GUI 的制作，将所实现的各项功能与图形用户界面结合，制作出了一款用户友好的音频处理小工具。通过这次项目，我们不仅将课内所学知识进行了实践，还扩展了我们的技术视野，学习了很多新知识。这次大作业不仅增强了我们的团队协作和项目管理能力，也为我们未来在信号处理领域的深入学习和应用打下了坚实的基础。展望未来，我们希望能在更多的实际项目中应用这些技术，不断提升自己的技能，并探索更多信号处理的新领域和新方法。