

西北工业大学计算机学院

《多媒体技术》实验报告

学 号： 10011904

姓 名： 李奇

实验时间： 2021. 11. 24

实验地点： 实验大楼 209-3

实验题目： Get to know audio

西北工业大学

2021 年 11 月

一、 实验目的及要求。

1. 了解并熟悉音频文件的基本特性。
2. 理解降采样等手段对音频文件的影响。
3. 熟悉浊音和辅音在共振峰上的不同以及上下文环境对他们的影响。
4. 了解压缩手段对文件大小和音质的影响。

二、 实验内容与操作步骤。

1. 观察给定的音乐文件（打击乐 弦乐）采样率，量化位数和通道数；同时观察编码格式是什么。

2. 录制一段自己的语音，内容为“音频语音处理技术真的很神奇啊”

3. 对上面音频文件（语音和音乐）进行降采样，降采样到 16K 和 8K，对比降采样前后的音质变化，写出主观感受。

4. 对降采样到 16K 的语音和音乐文件转换到频域，画出语谱图，观察最高频率范围是多少，并写出语音和音乐在语谱上特点差异。

5. 对 16K 的语音波形和语谱图进行进一步观察，写出轻音和浊音在波形和语谱上的差别；同时测量一下几个典型元音的基频（F0），观察这些不同元音在共振峰上的差别，再观察下同一个元音在不同上下文时在语谱上的差异。

6. 对两个音频文件使用 mp3 进行压缩，观察压缩前后文件大小和音质的变化。

三、 实验结果分析。

1. 观察给定的两个音乐文件（music_clip1&2，一个是打击乐片段，另一个是弦乐片段）的采样率，量化位数和通道数；同时观察编码格式是什么。这里使用 sox 工具的相应指令实现目标，结果如下图所示：

```
C:\Users\61060\Desktop\NPU-LQ\大三上\多媒体\LiQi_Audio\work1\音频>sox --i music_clip1.wav  
Input File      : 'music_clip1.wav'  
Channels        : 1  
Sample Rate     : 44100  
Precision       : 16-bit  
Duration        : 00:00:44.30 = 1953678 samples = 3322.58 CDDA sectors  
File Size       : 3.91M  
Bit Rate        : 707k  
Sample Encoding: 16-bit Signed Integer PCM
```

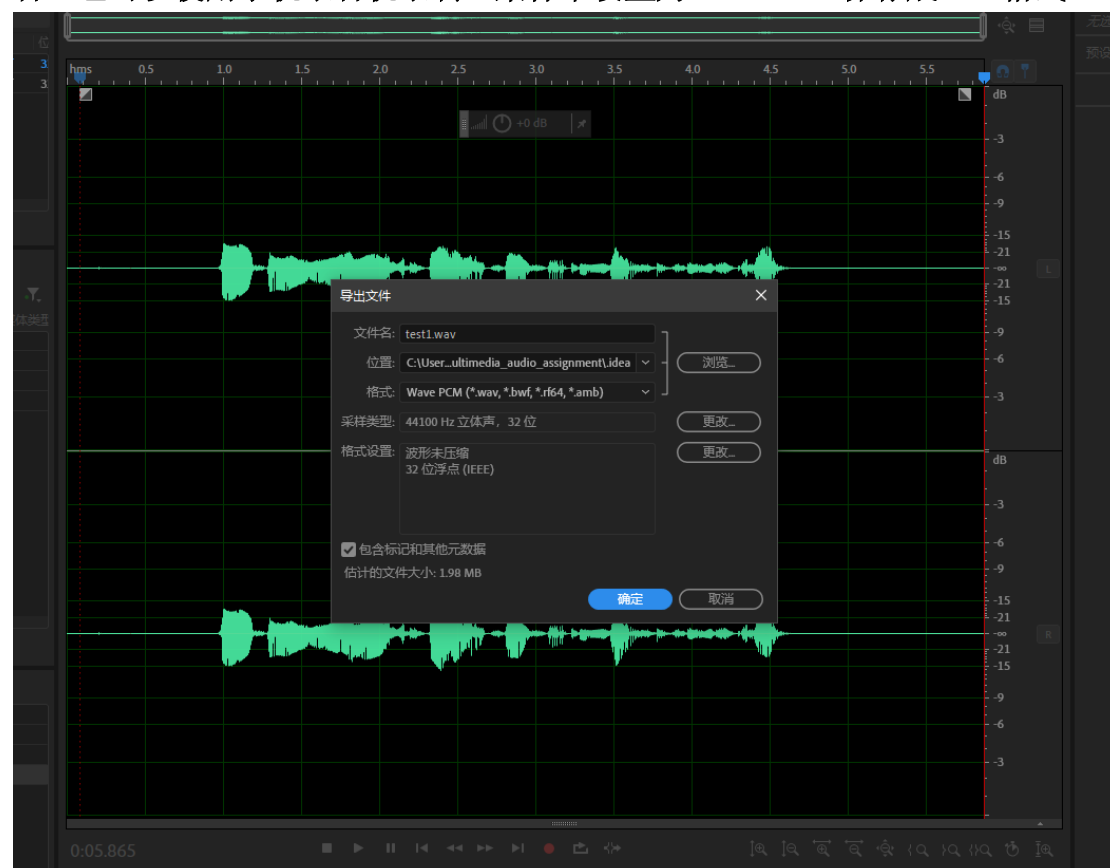
从图中可以看到，music_clip1 的采样率为 44.1kHz，量化位数为 16bit，通道数为 1，编码格式为 16-bit Signed Integer PCM。

```
C:\Users\61060\Desktop\NPU-LQ\大三上\多媒体\LiQi_Audio\work1\音频>sox --i music_clip2.wav

Input File      : 'music_clip2.wav'
Channels        : 1
Sample Rate     : 44100
Precision       : 16-bit
Duration        : 00:00:40.00 = 1764000 samples = 3000 CDDA sectors
File Size       : 3.53M
Bit Rate        : 707k
Sample Encoding : 16-bit Signed Integer PCM
```

从图中可以看到，music_clip1 的采样率为 44.1kHz，量化位数为 16bit，通道数为 1，编码格式为 16-bit Signed Integer PCM。

2. 录制一段自己的语音，内容为“音频语音处理技术真的很神奇啊”。可以用电脑录音机程序或者专业录音软件 audition 或 audacity（开源软件）录音，也可以使用手机录音机录制，采样率设置为 44.1KHz，保存成 wav 格式。



```
C:\Users\61060\Desktop\NPU-LQ\大三上\多媒体\LiQi_Audio\work1\音频>sox --i t1.wav
sox WARN wav: wave header missing extended part of fmt chunk

Input File      : 't1.wav'
Channels        : 2
Sample Rate     : 44100
Precision       : 25-bit
Duration        : 00:00:05.87 = 258656 samples = 439.891 CDDA sectors
File Size       : 2.08M
Bit Rate        : 2.83M
Sample Encoding : 32-bit Floating Point PCM
```

3. 用 Python 的 librosa 包对上面的音频文件（语音和音乐）进行读取并降采样到 16K 和 8K，对比降采样前后的音质变化，写出主观感受。对应的代码和音频信息如下图所示：

```
C:\Users\61060\Desktop\NPU-LQ\大三上\多媒体\LiQi_Audio\work1\音频>sox --i music_clip1_16.wav

1 import librosa
2 y, sr = librosa.load('music_clip1.wav', sr=None)
3 y_16k = librosa.resample(y, sr, 16000)
4 librosa.output.write_wav('music_clip1_16.wav', y_16k, 16000)
```

```
C:\Users\61060\Desktop\NPU-LQ\大三上\多媒体\LiQi_Audio\work1\音频>sox --i music_clip1_16.wav

Input File      : 'music_clip1_16.wav'
Channels        : 1
Sample Rate     : 16000
Precision       : 25-bit
Duration        : 00:00:44.30 = 708818 samples ~ 3322.58 CDDA sectors
File Size       : 2.84M
Bit Rate        : 512k
Sample Encoding : 32-bit Floating Point PCM
```

music_clip1_16.wav

```
C:\Users\61060\Desktop\NPU-LQ\大三上\多媒体\LiQi_Audio\work1\音频>sox --i music_clip1_8.wav

1 import librosa
2 y, sr = librosa.load('music_clip1.wav', sr=None)
3 y_8k = librosa.resample(y, sr, 8000)
4 librosa.output.write_wav('music_clip1_8.wav', y_8k, 8000)
```

```
C:\Users\61060\Desktop\NPU-LQ\大三上\多媒体\LiQi_Audio\work1\音频>sox --i music_clip1_8.wav

Input File      : 'music_clip1_8.wav'
Channels        : 1
Sample Rate     : 8000
Precision       : 25-bit
Duration        : 00:00:44.30 = 354409 samples ~ 3322.58 CDDA sectors
File Size       : 1.42M
Bit Rate        : 256k
Sample Encoding : 32-bit Floating Point PCM
```

music_clip1_8.wav

```
C:\Users\61060\Desktop\NPU-LQ\大三上\多媒体\LiQi_Audio\work1\音频>sox --i music_clip2_16.wav

1 import librosa
2 y, sr = librosa.load('music_clip2.wav', sr=None)
3 y_16k = librosa.resample(y, sr, 16000)
4 librosa.output.write_wav('music_clip2_16.wav', y_16k, 16000)
5 y_8k = librosa.resample(y, sr, 8000)
6 librosa.output.write_wav('music_clip2_8.wav', y_8k, 8000)
```

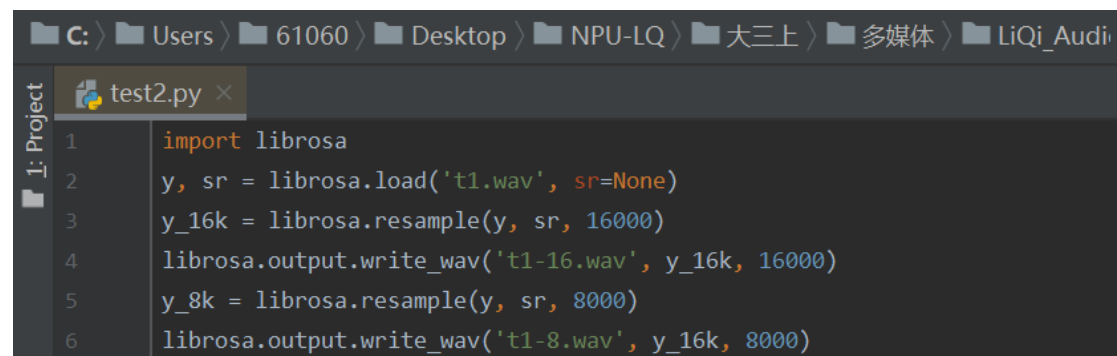
```
C:\Users\61060\Desktop\NPU-LQ\大三上\多媒体\LiQi_Audio\work1\音频>sox --i music_clip2_16.wav

Input File      : 'music_clip2_16.wav'
Channels        : 1
Sample Rate     : 16000
Precision       : 25-bit
Duration        : 00:00:40.00 = 640000 samples ~ 3000 CDDA sectors
File Size       : 2.56M
Bit Rate        : 512k
Sample Encoding : 32-bit Floating Point PCM
```

```
C:\Users\61060\Desktop\NPU-LQ\大三上\多媒体\LiQi_Audio\work1\音频>sox --i music_clip2_8.wav

Input File      : 'music_clip2_8.wav'
Channels        : 1
Sample Rate     : 8000
Precision       : 25-bit
Duration        : 00:00:40.00 = 320000 samples ~ 3000 CDDA sectors
File Size       : 1.28M
Bit Rate        : 256k
Sample Encoding : 32-bit Floating Point PCM
```

music_clip2_8.wav & music_clip2_16.wav



```
1 import librosa
2 y, sr = librosa.load('t1.wav', sr=None)
3 y_16k = librosa.resample(y, sr, 16000)
4 librosa.output.write_wav('t1-16.wav', y_16k, 16000)
5 y_8k = librosa.resample(y, sr, 8000)
6 librosa.output.write_wav('t1-8.wav', y_16k, 8000)
```

```
C:\Users\61060\Desktop\NPU-LQ\大三上\多媒体\LiQi_Audio\work1\音频>sox --i t1-16.wav

Input File      : 't1-16.wav'
Channels        : 1
Sample Rate     : 16000
Precision       : 25-bit
Duration        : 00:00:05.87 = 93844 samples ~ 439.894 CDDA sectors
File Size       : 375k
Bit Rate        : 512k
Sample Encoding : 32-bit Floating Point PCM
```

```
C:\Users\61060\Desktop\NPU-LQ\大三上\多媒体\LiQi_Audio\work1\音频>sox --i t1-8.wav

Input File      : 't1-8.wav'
Channels        : 1
Sample Rate     : 8000
Precision       : 25-bit
Duration        : 00:00:05.87 = 46922 samples ~ 439.894 CDDA sectors
File Size       : 188k
Bit Rate        : 256k
Sample Encoding : 32-bit Floating Point PCM
```

t1-8.wav & t1-16.wav

44.1k 时环境的杂音听起来十分清楚，立体感较强；降采样之后，明显感觉低频鼓点变弱，高频不够通透，细节丢失，底噪变大，声音变得越来越模糊，声音逐渐变质，有种对讲机说话的感觉。总体来说音质下降比较明显。

4. 通过使用 python 里的 matplotlib.pyplot 对降采样到 16K 的语音和音乐文件转换到频域，画出语谱图，观察最高频率范围是多少，并写出语音和两段音乐在语谱上特点差异。实现代码及对应结果图如下所示：

```
import librosa
import numpy as np
import matplotlib.pyplot as plt
import os
import wave

# 读入音频。
path = "C:\\Users\\61060\\Desktop\\NPU-LQ\\大三上\\多媒体\\LiQi_Audio\\work1\\音频"
name = 'music_clip1.wav'
filename = os.path.join(path, name)

# 打开语音文件。
f = wave.open(filename, 'rb')
# 得到语音参数
params = f.getparams()
nchannels, sampwidth, framerate, nframes = params[:4]
# -----#
# 将字符串格式的数据转成 int 型
print("reading wav file.....")
strData = f.readframes(nframes)
waveData = np.fromstring(strData, dtype=np.short)
# 归一化
waveData = waveData * 1.0 / max(abs(waveData))
# 将音频信号规整乘每行一路通道信号的格式，即该矩阵一行为一个通道的采样点，共 nchannels 行
waveData = np.reshape(waveData, [nframes, nchannels]).T # .T 表示转置
f.close() # 关闭文件
print("file is closed!")
# -----#
'''绘制语音波形'''
print("plotting signal wave...")
time = np.arange(0, nframes) * (1.0 / framerate) # 计算时间
time = np.reshape(time, [nframes, 1]).T
plt.plot(time[0, :nframes], waveData[0, :nframes], c="b")
plt.xlabel("time")
plt.ylabel("amplitude")
plt.title("Original wave")
plt.show()

# -----#
```

```

'''
    绘制频谱
    1. 求出帧长、帧叠点数。且 FFT 点数等于每帧点数（即不补零）
    2. 绘制语谱图
'''

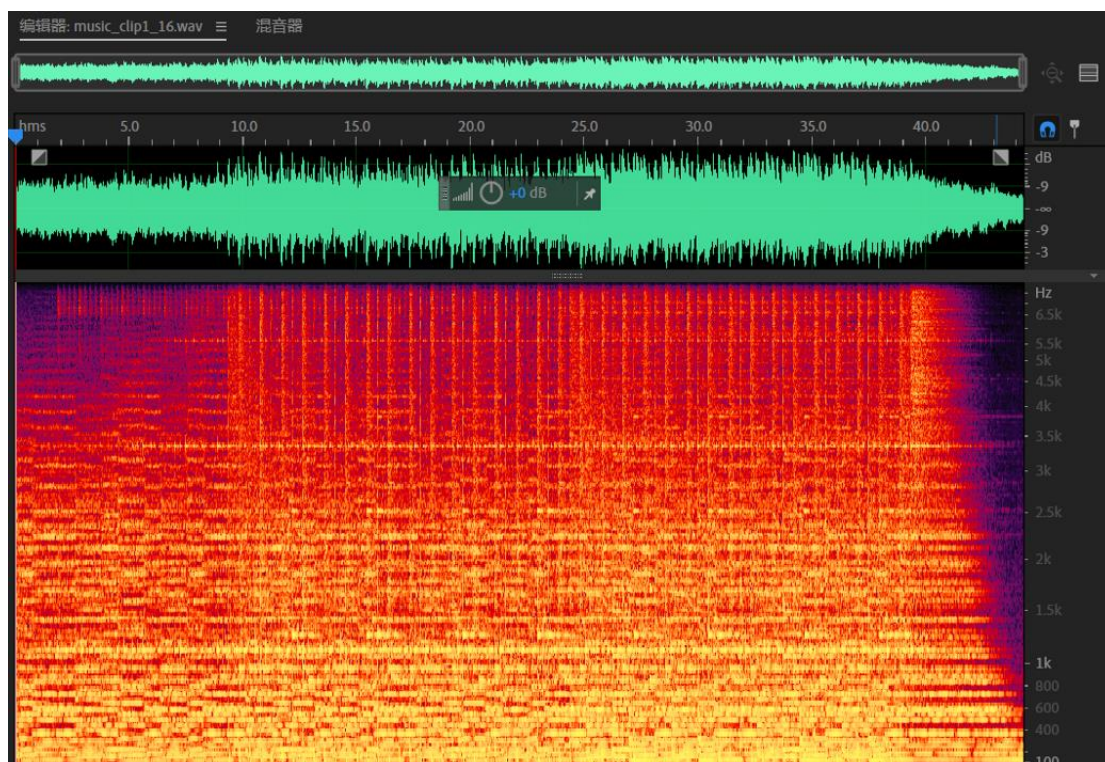
print("plotting spectrogram...")
framelength = 0.025 # 帧长 20~30ms
framesize = framelength * framerate # 每帧点数 N = t*fs, 通常情况下值为 256 或 512, 要与 NFFT 相等\
# 而 NFFT 最好取 2 的整数次方, 即 framesize 最好取的整数次方

# 找到与当前 framesize 最接近的 2 的正整数次方
nfftdict = {}
lists = [32, 64, 128, 256, 512, 1024]
for i in lists:
    nfftdict[i] = abs(framesize - i)
sortlist = sorted(nfftdict.items(), key=lambda x: x[1]) # 按与当前 framesize 差值升序排列
framesize = int(sortlist[0][0]) # 取最接近当前 framesize 的那个 2 的正整数次方值为新的 framesize

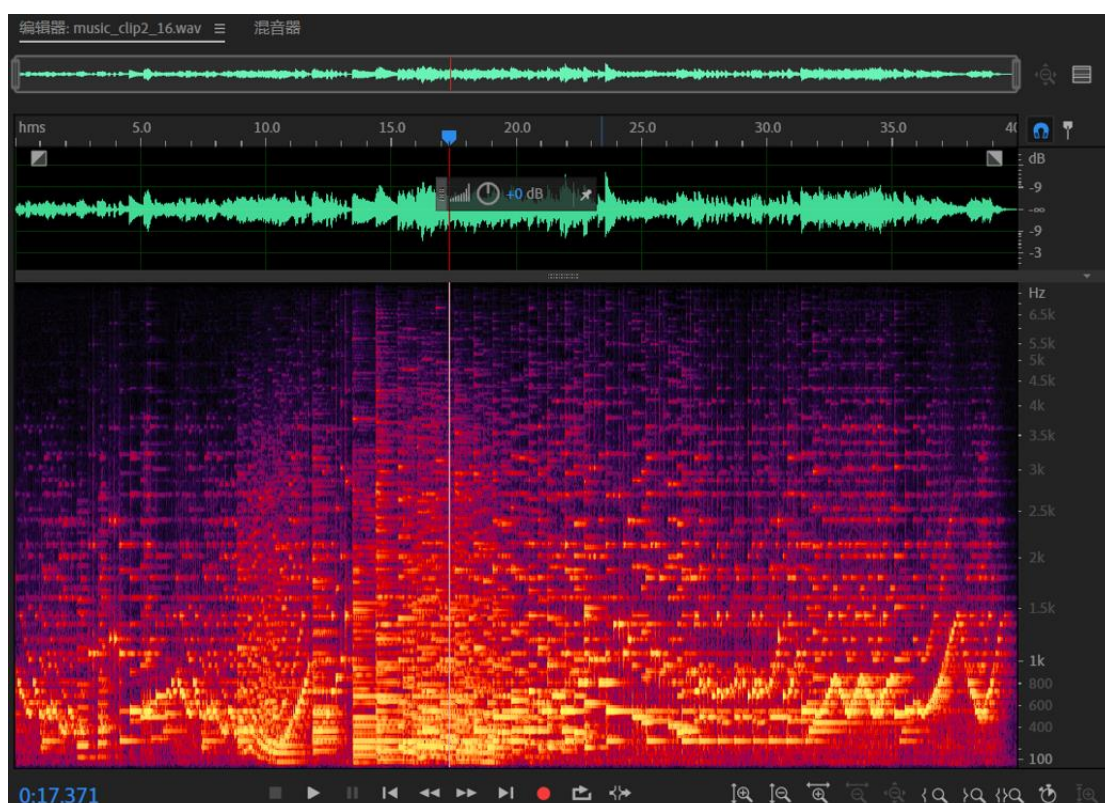
NFFT = framesize # NFFT 必须与时域的点数 framesize 相等, 即不补零的 FFT
overlapSize = 1.0 / 3 * framesize # 重叠部分采样点数 overlapSize 约为每帧点数的 1/3~1/2
overlapSize = int(round(overlapSize)) # 取整
print("帧长为{}, 帧叠为{}, 傅里叶变换点数为{}".format(framesize, overlapSize, NFFT))
spectrum, freqs, ts, fig = plt.specgram(waveData[0], NFFT=NFFT,
Fs=framerate, window=np.hanning(M=framesize),
noverlap=overlapSize, mode='default',
scale_by_freq=True, sides='default',
scale='dB', xextent=None) # 绘制频谱图

plt.ylabel('Frequency')
plt.xlabel('Time')
plt.title("Spectrogram")
plt.show()

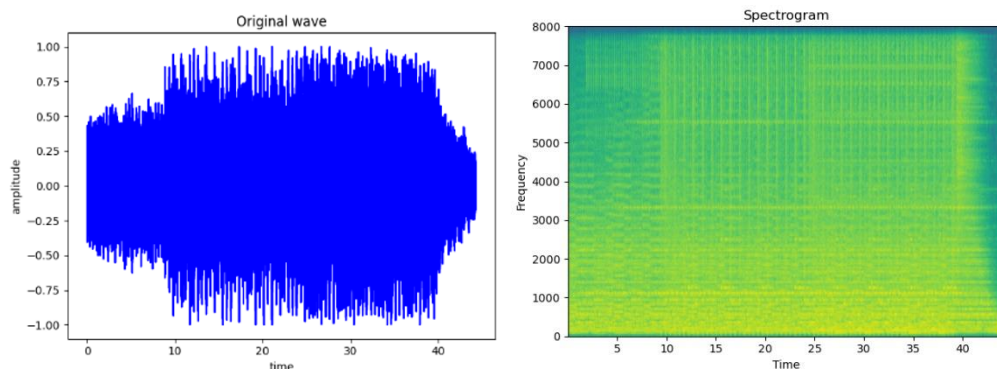
```

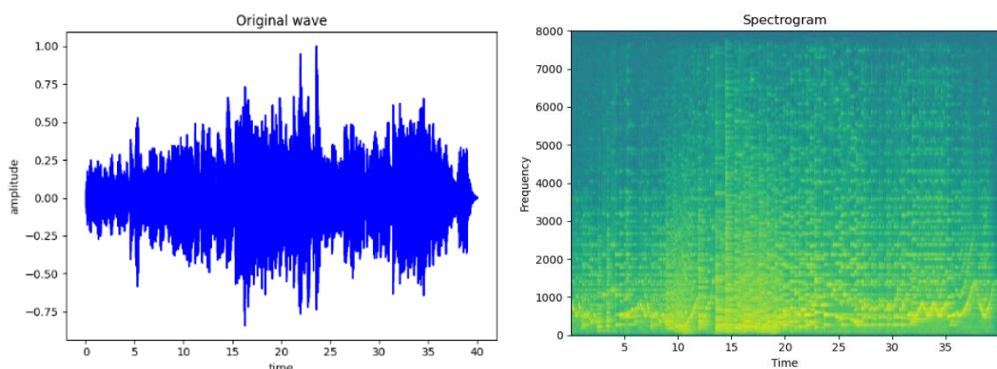
music_clip1_16. wav



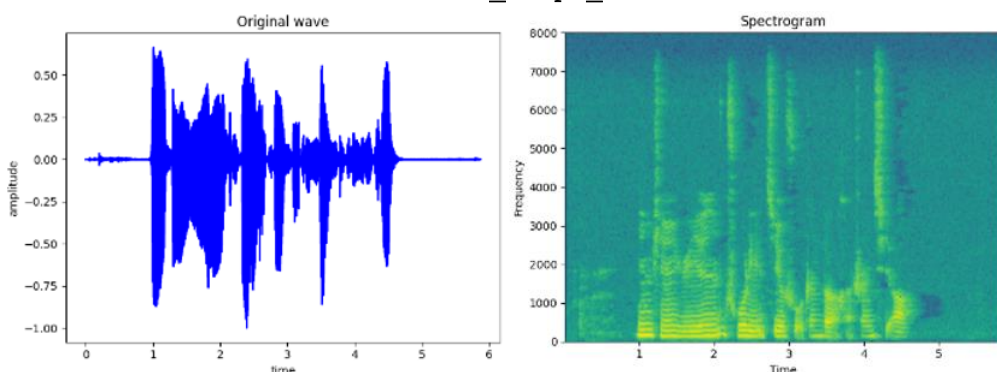
music_clip2_16. wav



`music_clip1_16.wav`



`music_clip2_16.wav`



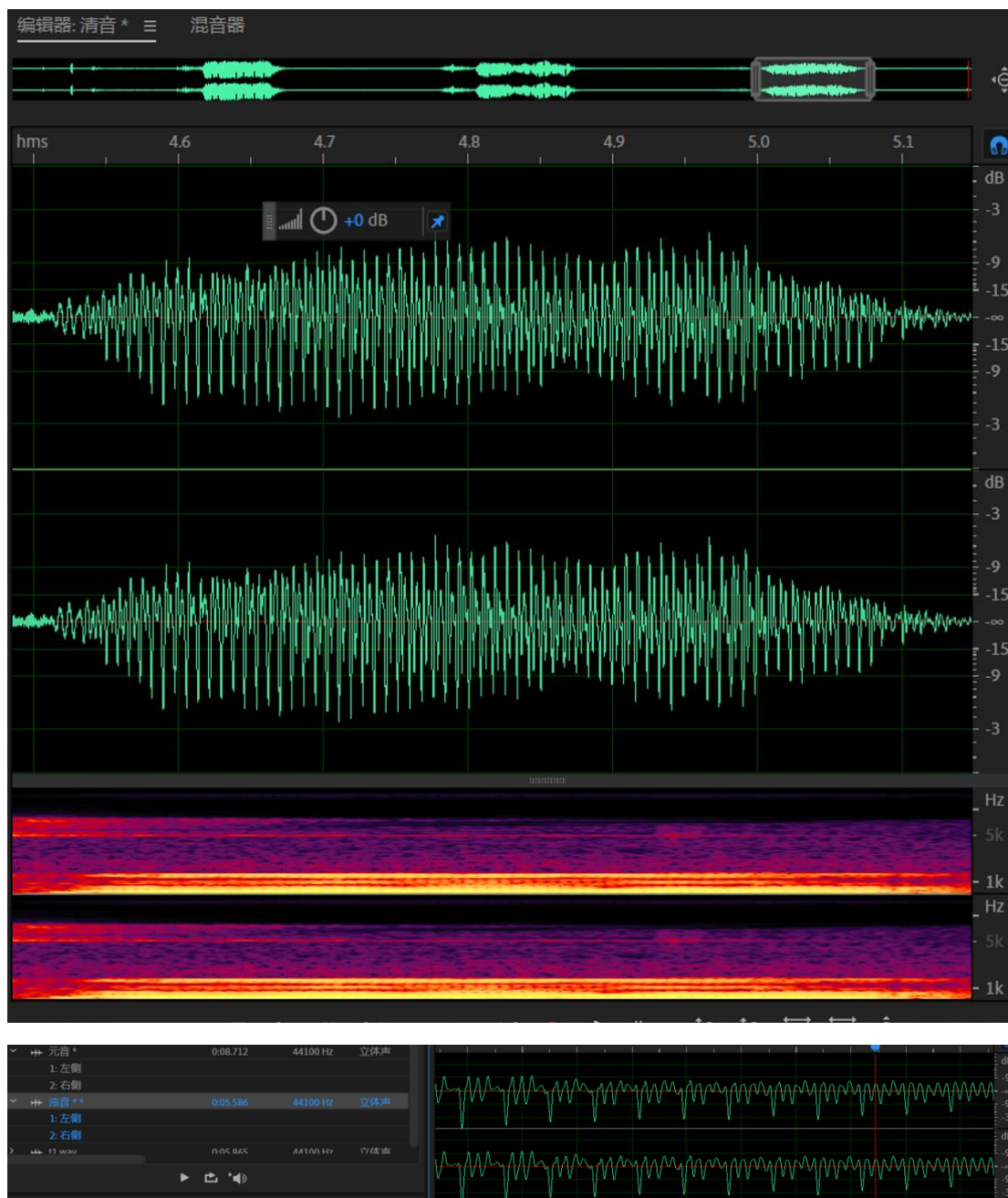
`t1-16.wav`

从图中可以看出，`music_clip1` 在降采样到 16K 后最高频率在 7800Hz 左右，`music_clip2` 在降采样到 16K 后最高频率在 8000Hz 左右，自己录制的音频降采样到 16K 后最高频率在 7600 左右，都符合奈奎斯特采样定律。

同时通过对比可以看出，两段音乐的频率变化更加均匀有规律，具有连续的变化，并且低中高频区都有均匀的分布。而自己的语音频率变化是断开的，能看出每个字的发音区域，高频区和低频区都有分布，而中频区就也有断开的地方，并不是均匀的分布。

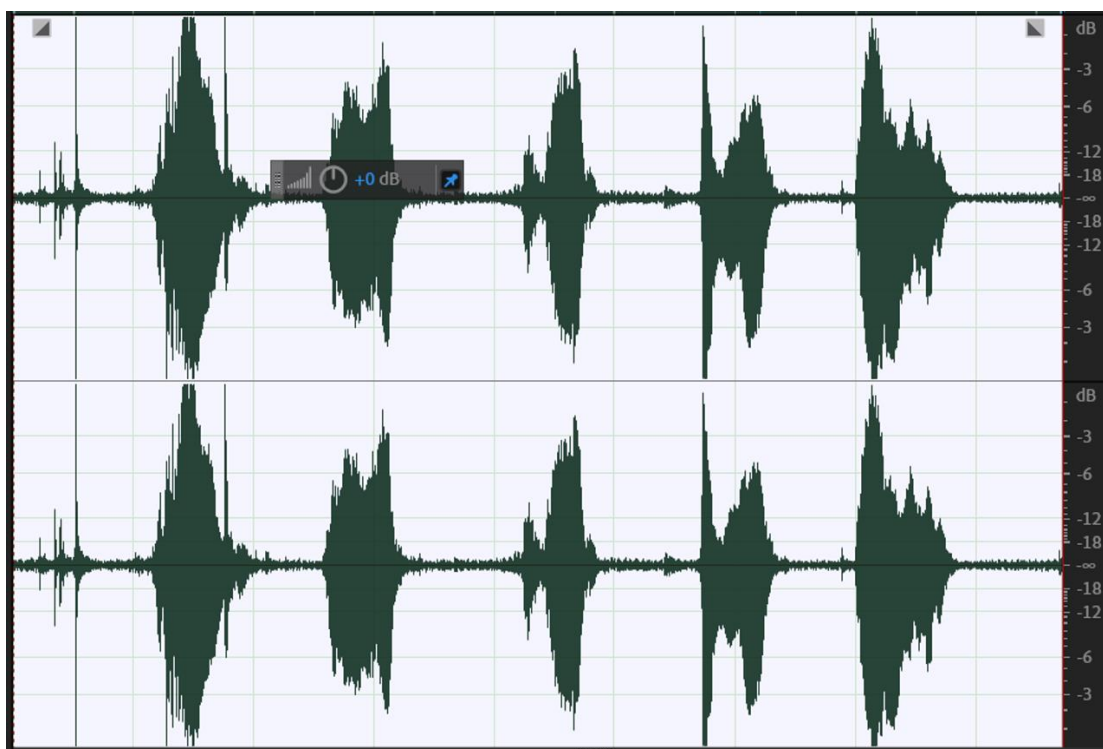
5. 对 16K 的语音波形和语谱图进行进一步观察，写出轻音和浊音在波形和语谱上的差别；同时测量一下几个典型元音的基频（F0），观察这些不同元音在共振峰上的差别，再观察下同一个元音在不同上下文时在语谱上的差异。

录制了清音.wav 和浊音.wav 文件，相应的信息如下图所示：

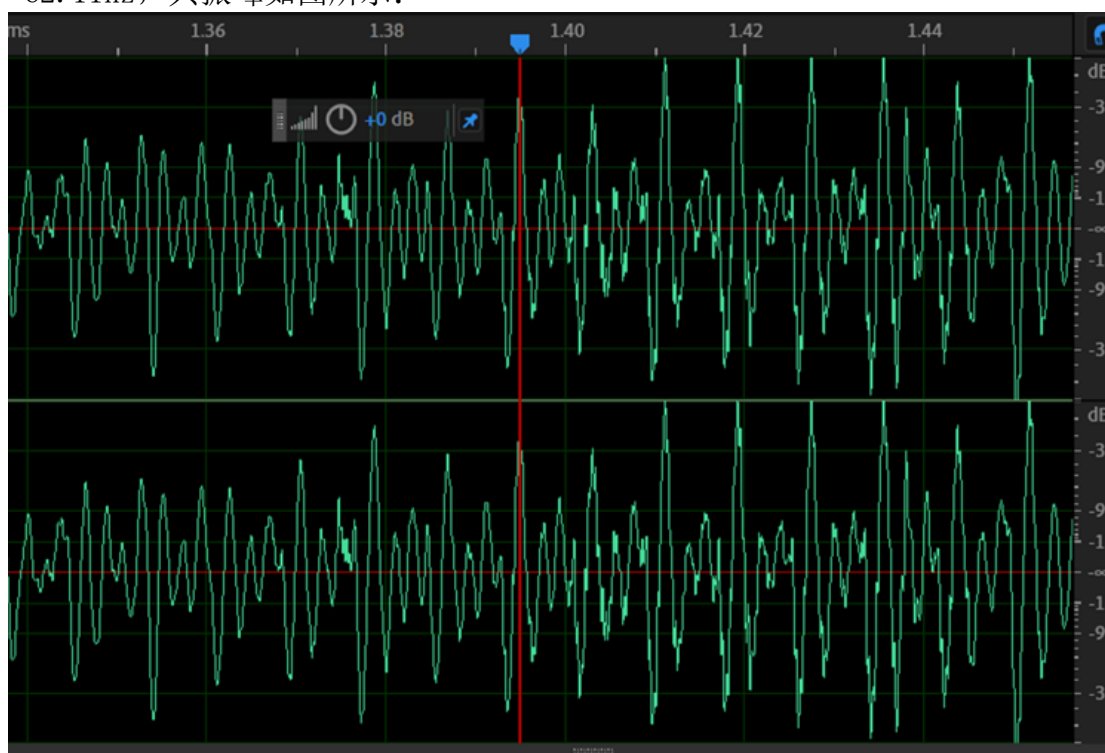


通过观察可以发现，浊音信号的能量集中在低频区、清音信号的能量集中在高频区。不同元音的共振峰与频带宽度不同。且清音的波形无规律，浊音的波形呈周期性。音乐的波形图曲线变幻幅度大，且频率高，人声的变幻幅度慢，且频率低。

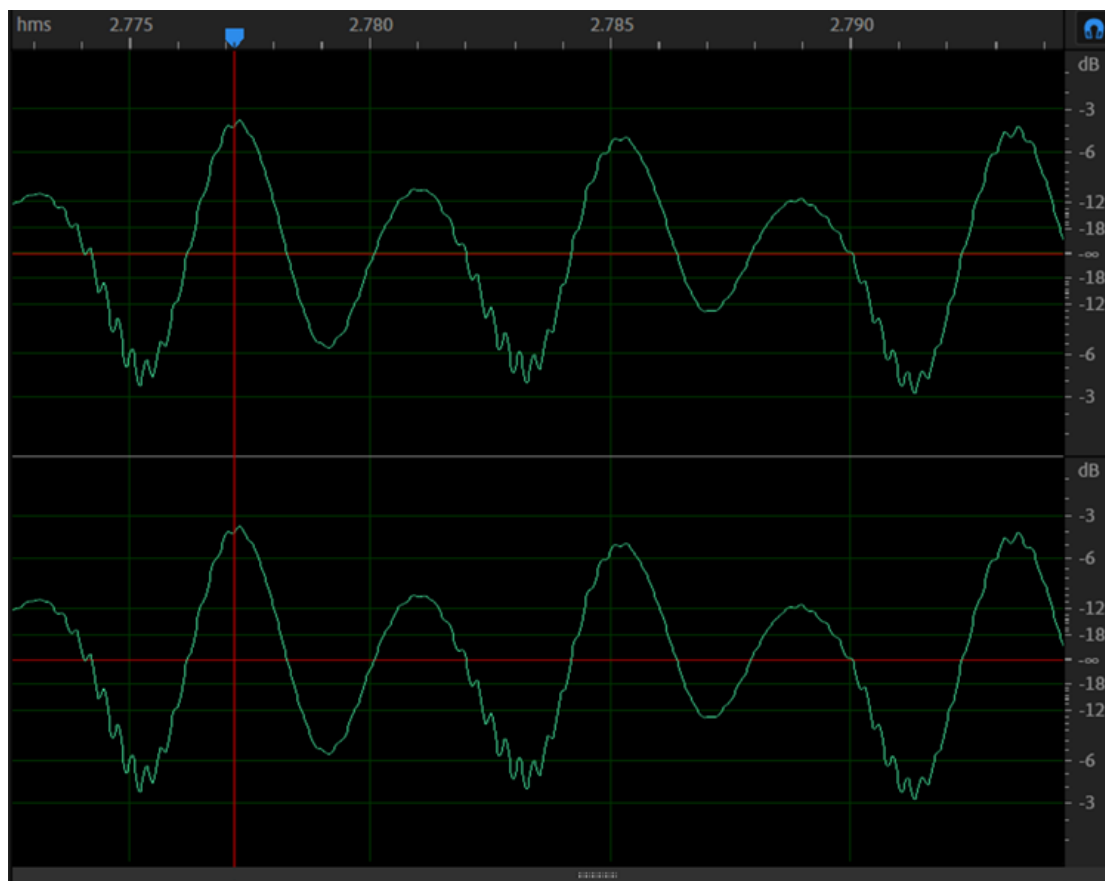
下面分析各个元音的波形图和语谱图信息，相应结果如下图所示，分别对应元音 a、e、i、o、u：



对于元音 a，相应信息如下，计算可知基频 = $1 / (1.3943 - 1.3782)$
=62.11hz，共振峰如图所示：



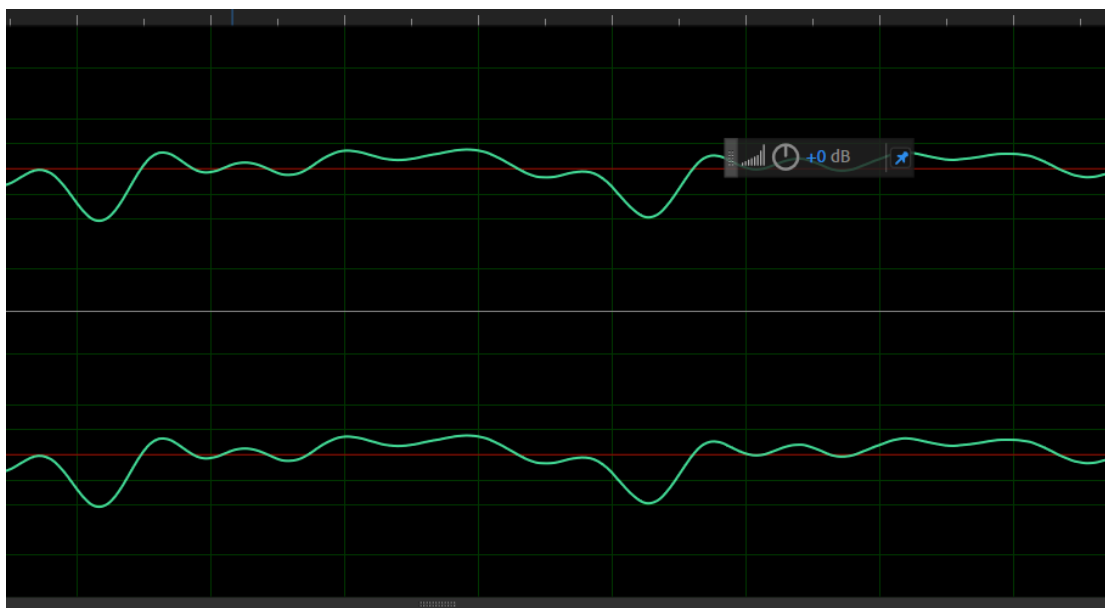
对于元音 e，相应信息如下，计算可知基频 = $1 / (2.7853 - 2.7772)$
=123.47hz，共振峰如图所示：



对于元音 i，相应信息如下，计算可知基频 = $1 / (4.4823 - 4.4739) = 119.05\text{Hz}$ ，共振峰如图所示：



对于元音 o，相应信息如下，计算可知基频 = $1 / (5.8884 - 5.8802) = 121.95\text{Hz}$ ，共振峰如图所示：

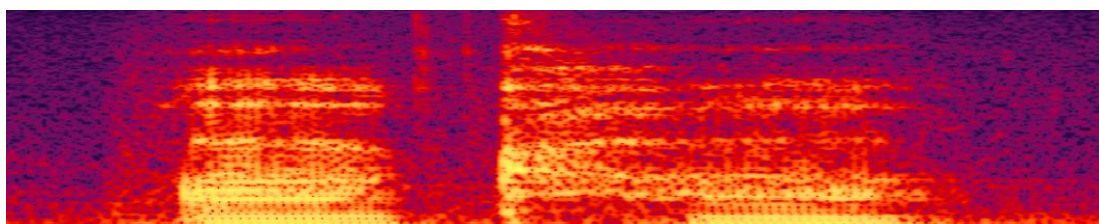


对于元音 u，相应信息如下，计算可知基频 = $1 / (7.1154 - 7.1075)$
=126.58hz，共振峰如图所示：











综上分析共振峰的差别，通过分析语谱图可知，谐波中颜色比同时刻附近其他横条纹颜色更深的条纹表示共振峰，当出现颜色比局部其他部分颜色深的多条条纹时，代表着出现各次共振峰。不同元音共振峰差别较大，相比较而言，a、i、u 的共振峰数量更多一些，而 e 和 o 只在低频部分存在共振峰，除此之外，各个元音共振峰的走势也不尽相同，a、e、i、o 的共振峰呈下凹状，u 的共振峰呈上凸状。

下面分析同一元音在不同的上下文语境中语谱的不同，这里选择了元音 a，其在清音和浊音语境下的语谱图分别如下所示：

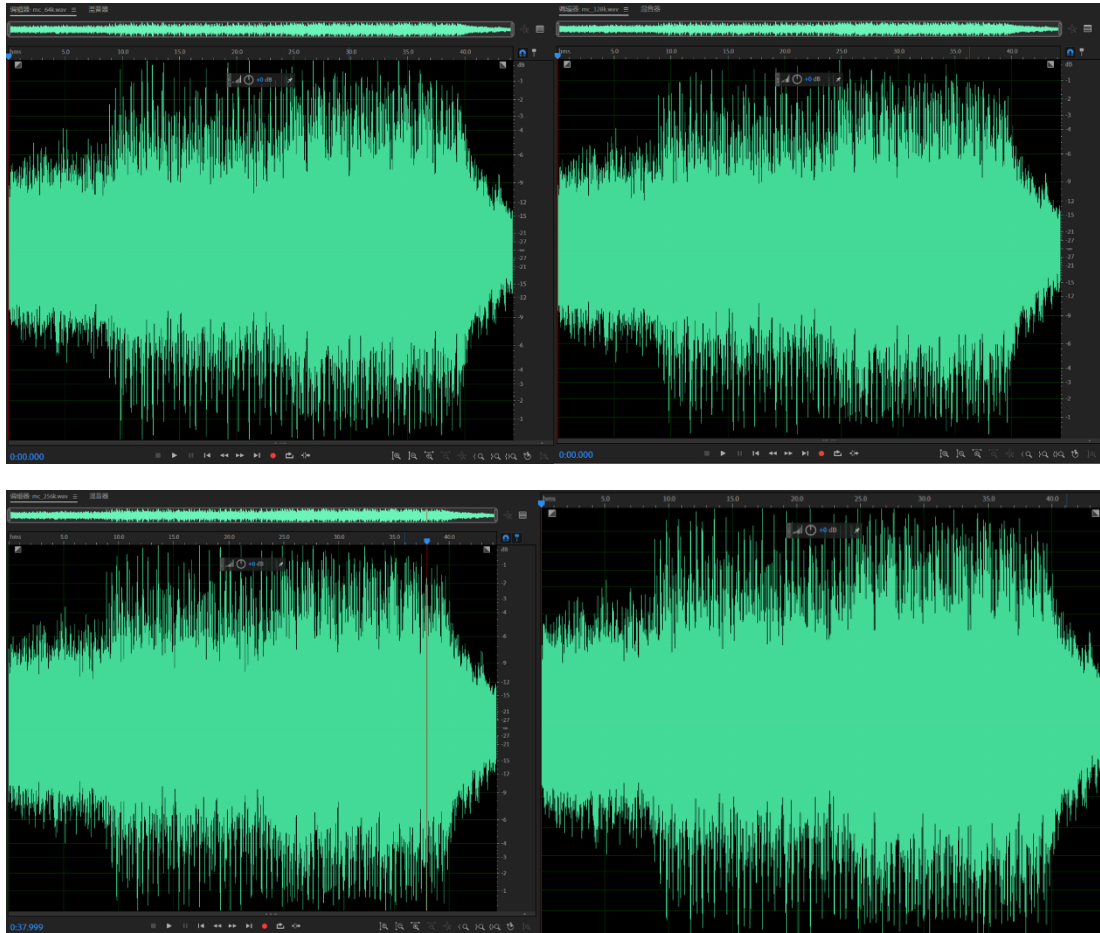



```
C:\Users\61060\Desktop>NNFL-LQ(天工)多媒体\LiQL_Audio\work1\音频\Ffmpeg -i music.clipl.wav -vn -ar 44.1k -ac 2 -ab 256k -f mp3 mc_256k.wav
Ffmpeg version 2021.11.25 git 5d22577d9a full build www.gyan.dev Copyright (c) 2000-2021 the FFMpeg developers
Built with gcc 11.2.0 (Rev2), Built by MSYS2 project
configuration: enable_av1 enable_version3 enable_static disable_x3threads disable_outdetect enable_fontconfig enable_iconv enable_gnutls enable_librpm2 enable_mn
enable_zma enable_lzma enable_libriscv enable_zlib enable_librist enable_librt enable_lmbzq enable_vixsynth enable_libray enable_lbcaca enable_sdl2 e
enable_lbvadv enable_lbvadv2 enable_lbvadv3d enable_lbvzvi enable_lbvavie enable_lbvstavl enable_lbwepb enable_lbx264 enable_lbx265 enable_lbvax25 enable_l
lhxv enable_lbvadv enable_lbvopenjpeg enable_lbvpc enable_lbvpc enable_lbvpc enable_freob enable_lbvpc enable_lbvpc enable_lbvpc enable_lbvpc enable_lbvpc enabl
enable_cuda enable_cuda_nvml enable_cudnn enable_ffmpeg enable_nvcv enable_dsvla enable_lbtz enable_lbtz enable_lbtz enable_lbtz enable_lbtz enable_lbtz enabl
enable_lbpheco enable_openc1 enable_lbcdio enable_lbgme enable_lbmddplug enable_lbpemmp enable_lbpemcp enable_lbpemcp enable_lbpemcp enable_lbpemcp enable_lbpemcp enabl
enable_lbvlow enable_lbvlow enable_lbvlow enable_lbvlow enable_lbvlow enable_lbvlow enable_lbvlow enable_lbvlow enable_lbvlow enable_lbvlow enable_lbvlow enable_lbvlow enabl
enable_lbvlow enable_lbvlow enable_lbvlow enable_lbvlow enable_lbvlow enable_lbvlow enable_lbvlow enable_lbvlow enable_lbvlow enable_lbvlow enable_lbvlow enable_lbvlow enabl
libavutil 57. 9.101 / 57. 9.101
libavcodec 59. 3.101 / 59. 3.101
libavformat 59. 9.102 / 59. 9.102
libavdevice 59. 0.101 / 59. 0.101
libavfilter 8. 17.100 / 8. 17.100
libbscale 6. 1.100 / 6. 1.100
libswresample 4. 0.100 / 4. 0.100
libpostproc 56. 0.100 / 56. 0.100
Guessed Channel Layout for Input Stream #0.0 : mono
Input #0, wav, from 'music.clipl.wav':
Duration: 00:00:44.30, bitrate: 706 kb/s
Stream #0:0 Audio: pcm_s16le ([1][0][0] [0] / 0x0001). 44100 Hz, mono, s16, 705 kb/s
Stream mapping:
Stream #0:0 --> #0:0 (pcm_s16le (native)) --> #0:3 (libmp3lame)
Press [q] to stop, [?] for help.
Output #0, mp3, to 'mc_256k.wav':
Metadata:
TSSE : LavF59.9.102
Stream #0:0 Audio: mp3, 44100 Hz, stereo, s16p, 256 kb/s
Metadata:
encoder : LavF59.13.101 libmp3lame
size= 1386Kbit time=00:00:44.30 bitrate= 256.3Kbits/s speed=30.5x
video 0KB audio 1385K subtitle 0KB other streams 0KB global headers 0KB muxing overhead: 0.061965%
```

```
F:\Users\c01600\Desktop\NPN-1\9天二上多媒体\1\1.Q音频\work\音效\ffmpeg -i music.clip1.wav -vn -ar 44.1k -ac 2 -ab 1024K -f mp3 mc_1024k.wav
ffmpeg version 2021.11.25 git-5222577dfe-full_build_wmcs.gyan.dev Copyright (c) 2000-2021 The FFmpeg Developers
built with gcc 11.2.0 (Rev2), built by MSYS2 project
configuration: enable_avcodec=enable_avdevice=enable_avfilter=enable_avformat=enable_avutil=enable_bzlib=enable_curl=enable_ebur128=enable_fdk_aac=enable_gnutls=enable_iconv=enable_lcms2=enable_libass=enable_libbluray=enable_libcdio=enable_libcdda=enable_libcddb=enable_libcelt=enable_libchromaprint=enable_libclang=enable_libcoq=enable_libcurl=enable_libdc1394=enable_libdrm=enable_libfdk_aac=enable_libflite=enable_libfluidsynth=enable_libgsm=enable_liblame=enable_libmodplug=enable_libmp3lame=enable_libopenjpeg=enable_libopus=enable_libpulseaudio=enable_librtmp=enable_librubberband=enable_libsoxr=enable_chromaprint
enabled features: avcodec, avdevice, avfilter, avformat, avutil, bzlib, curl, ebur128, fdk-aac, gnutls, iconv, lcms2, libass, libbluray, libcdio, libcdda, libcddb, libcelt, libchromaprint, libclang, libcoq, libcurl, libdc1394, libdrm, libfdk-aac, libflite, libfluidsynth, libgsm, liblame, libmodplug, libmp3lame, libopenjpeg, libopus, libpulseaudio, librtmp, librubberband, libsoxr, chromaprint
disabled features: none
Input #0, wav, from music.clip1.wav:
Duration: 00:00:44.30, bitrate: 706 kb/s
Stream #0:0 Audio: pcm_s16le ([1][0][0] / 0x0001), 44100 Hz, mono, s16, 705 kb/s
Stream mappings:
Stream #0:0 --> #0:0 (pcm_s16le (native)) --> #0:3 (libmp3lame)
Press [q] to stop, [?] for help
Output #0, mp3, to 'mc_1024k.wav':
Metadata:
TSSE
Stream #0:0 Audio: mp3, 44100 Hz, stereo, s16p, 1024 kb/s
Metadata:
encoder
LAME3.100r13101 libmp3lame
size= 1733KB time=00:00:44.30 bitrate= 320.4kbits/s speed=30.5X
video:0KB audio:1732KB subtitle:0KB other streams:0KB global headers:0KB muxing overhead: 0.061358%
```

	mc_64k.wav		mc_128k.wav
文件类型:	波形声音 (.wav)	文件类型:	波形声音 (.wav)
打开方式:	 Windows Media Player 更改(C)...	打开方式:	 Windows Media Player 更改(C)...
位置:	C:\Users\61060\Desktop\NPU-LQ\大三上\多媒体\LiQi	位置:	C:\Users\61060\Desktop\NPU-LQ\大三上\多媒体\LiQi
大小:	346 KB (354,890 字节)	大小:	693 KB (709,737 字节)
占用空间:	348 KB (356,352 字节)	占用空间:	696 KB (712,704 字节)
	mc_256k.wav		mc_1024k.wav
文件类型:	波形声音 (.wav)	文件类型:	波形声音 (.wav)
打开方式:	 Windows Media Player 更改(C)...	打开方式:	 Windows Media Player 更改(C)...
位置:	C:\Users\61060\Desktop\NPU-LQ\大三上\多媒体\LiQi	位置:	C:\Users\61060\Desktop\NPU-LQ\大三上\多媒体\LiQi
大小:	1.35 MB (1,419,432 字节)	大小:	1.69 MB (1,774,279 字节)
占用空间:	1.35 MB (1,421,312 字节)	占用空间:	1.69 MB (1,777,664 字节)

通过观察分析可以得出，采用 MP3 压缩后，文件的大小和音质降低；随着码率上升，文件大小越来越大，码率逐渐升高，音质也越来越好。打开音频试听后发现码率高的音频对于音频的细节保留较好，码率低的音频一些细节出现丢失现象。



四、实验中存在问题和改进。

下载 librosa 时遇到的问题：下载 librosa 后无法 import，经过查阅资料得知，问题在于 numba 和 librosa 版本不兼容，解决方案如下：

```
sudo pip3 install numba == 0.42.0
sudo pip3 install librosa == 0.6.0
import librosa
```