江 西 理 工 大 学

机器学习 实验报告

实验名称 实验一 语音采集与读写实验

日 期 2023.3.02 专 业 智能科学与技术 班 级 智科 201 班

实验人 吴佳龙 学 号 2420203236 同组人

一、实验目的

1、了解 Python 采集语音信号的原理及常用命令。

2、熟练掌握基于 Python 的语音文件的创建、读写等基本操作。

3、掌握 matplotlib 来显示语音信号波形及基本的标注方法。

二、实验原理

1、语音信号特点

通过对大量语音信号的观察和分析发现，语音信号主要有下面两个特点:

1)在频域内，语音信号的频谱分量主要集中在 300~3400Hz 的范围内。利用这个点，可以用一个防混叠的带通滤波器将此范围内的语音信号频率分出，然后按 8kHz 的采中率对语音信号进行采样，就可以得到离散的语音信号。

2)在时域内，语音信号具有“短时性”的特点，即在总体上，语音信号的特征是随时间而变化的，但在一段较短的时间间隔内，语音信号保持平稳。在浊音段表现出周期信号的特征，在清音段表现出随机噪声的特征。

在 Windows 环境下，学生可以使用 Windows 自带的录音机录制语音文件，图 2-1 是基于 PC 的语音信号采集过程，声卡可以完成语音波形的 A/D 转换，获得 WAV 文件。通过 Windows 录制的语音信号，一方面可以为后续实验储备原始语音，另一方面可以与通过其他方式录制的语音进行比对。

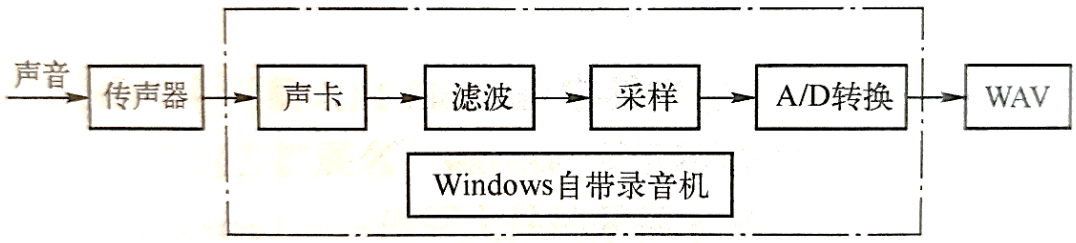


图 2-1 PC 语音信号采集原理图

三、实验设备及环境

1、设备：笔记本、台式电脑

2、实验环境：Windows 7、Windows10 或 Linux 操作系统

MATLAB/Python3.6+，Anaconda3，Pycharm2017+

四、实验内容与步骤

1、编写 python 程序利用 pyaudio 实现录制语音信号“你好，智科”，并保存为 audio_01.wav 文件，要求采样频率为 32000Hz，采样精度为 16bit。

2、使用 matplotlib 和 librosa 绘制语音文件的时域波形图，要求：横纵坐标带有标注，横轴单位为秒（s），纵轴显示归一化后的值。

3、使用 pyaudio 和 wave 函数包播放录制的语音信号，并改变播放的采样频率体验不同的效果。

五、实验注意事项

1、实验中应遵守实验室实验规则，爱护实验室设备，保持卫生清洁；各种实验仪器设备在观察使用后放归原处，不得损坏或随意放置。

六、思考题

1、语音信号的常见文件格式有哪些？

WAV（Waveform Audio File Format）：WAV 是 Windows 操作系统中常见的一种无损音频格式，可以储存高质量的音频数据。它是由 Microsoft 和 IBM 共同开发的文件格式，支持多种音频编码格式。

MP3（MPEG Audio Layer-3）：MP3 是一种有损压缩音频格式，可以将音频文件压缩到原始文件大小的一般左右，适合在网络上传输和存储。MP3 广泛应用于音乐播放器和流媒体平台中。

AAC（Advanced Audio Coding）：AAC 是一种有损压缩音频格式，与 MP3 相比，具有更高的音频质量和更小的文件大小。AAC 广泛应用于数字音频广播、数字电视和在线音乐服务中。

FLAC（Free Lossless Audio Codec）：FLAC 是一种无损压缩音频格式，可以保留原始音频数据的完整性，但文件大小相对较大。FLAC 广泛应用于音乐存储和音频制作中。

OGG（Ogg Vorbis）：OGG 是一种开源的无损压缩音频格式，具有高质量、低延迟和低码率等特点。OGG 广泛应用于音频和视频编码、在线游戏和网络会议中。

AIFF（Audio Interchange File Format）：AIFF 是一种无损音频格式，广泛应用于音频制作和录音工作中，也支持多种音频编码格式。

2、为保证信号不失真，对采样频率有何要求？

在数字信号处理中，为了保证信号不失真，采样频率必须满足采样定理，也称为奈奎斯特采样定理。采样定理指出，为了能够准确地重构出原始信号，采样频率必须高于原始信号中最高频率的两倍。

具体而言，若原始信号中的最高频率为 f_{max}，则采样频率 f_s 必须满足：

$$f_s > 2f_{max}$$

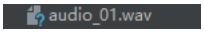
否则，信号就会失真。失真的原因是采样频率不足以捕捉到原始信号中的高频分量，从而导致高频分量被误解释成低频分量。这种误解释被称为混叠效应。因此，为了保证信号不失真，应当选择足够高的采样频率。在实际应用中，通常会选择比最高频率高一个数量级的采样频率，以确保信号的质量和准确性。

七、实验程序及结果展示

1)
```
#-coding:utf-8-*-
import pyaudio
import wave

# 设置录音参数
CHUNK = 1024
FORMAT = pyaudio.paInt16
CHANNELS = 1
RATE = 32000
RECORD_SECONDS = 5
WAVE_OUTPUT_FILENAME = "audio_01.wav"

# 初始化 PyAudio
audio = pyaudio.PyAudio()

# 开始录音
stream = audio.open(format=FORMAT, channels=CHANNELS,
                    rate=RATE, input=True,
                    frames_per_buffer=CHUNK)

print("Recording...")

frames = []

for i in range(0, int(RATE / CHUNK * RECORD_SECONDS)):
    data = stream.read(CHUNK)
```

```
        frames.append(data)

print("Finished recording.")

# 停止录音并关闭流
stream.stop_stream()
stream.close()
audio.terminate()

# 将录音结果保存到 WAV 文件中
wf = wave.open(WAVE_OUTPUT_FILENAME, 'wb')
wf.setnchannels(CHANNELS)
wf.setsampwidth(audio.get_sample_size(FORMAT))
wf.setframerate(RATE)
wf.writeframes(b''.join(frames))
wf.close()
```

结果图



实验结果分析

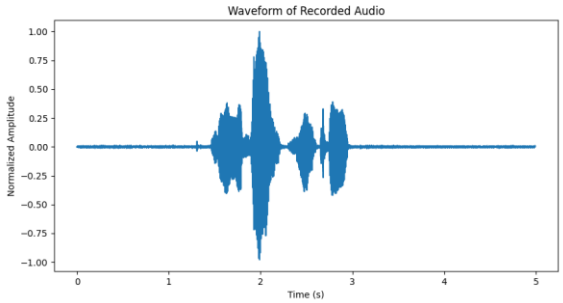
生成了音频文件 audio_01.wav

2）

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

# 读取 WAV 文件
wave_file = wave.open("audio_01.wav", 'rb')

# 获取音频参数
nchannels = wave_file.getnchannels()
sampwidth = wave_file.getsampwidth()
framerate = wave_file.getframerate()
nframes = wave_file.getnframes()

# 读取音频数据
data = wave_file.readframes(nframes)
wave_file.close()

# 转换音频数据为 numpy 数组
audio_signal = np.frombuffer(data, dtype=np.int16)
```

```
# 归一化音频数据
audio_signal = audio_signal / np.max(np.abs(audio_signal))

# 计算时间轴
time = np.arange(0, nframes/framerate, 1/framerate)

# 绘制时域波形图
plt.figure(figsize=(10, 5))
plt.plot(time, audio_signal)
plt.xlabel('Time (s)')
plt.ylabel('Normalized Amplitude')
plt.title('Waveform of Recorded Audio')
plt.show()
```

结果图



实验结果分析

通过音频文件得到了时域波形图

3）

```
import wave
import pyaudio

# 打开 WAV 文件
wave_file = wave.open("audio_01.wav", 'rb')

# 获取音频参数
nchannels = wave_file.getnchannels()
sampwidth = wave_file.getsampwidth()
framerate = wave_file.getframerate()
nframes = wave_file.getnframes()

# 读取音频数据
data = wave_file.readframes(nframes)
wave_file.close()

# 创建 PyAudio 对象
p = pyaudio.PyAudio()
```

```
# 打开音频输出流
stream = p.open(format=p.get_format_from_width(sampwidth),
                channels=nchannels,
                rate=framerate,
                output=True)

# 播放原始音频数据
stream.write(data)

# 改变采样频率为 48000Hz
new_framerate = 48000
new_data = wave.open('audio_01_new.wav', 'wb')
new_data.setn
```