哈爾濱工業大學

实验报告

实 验（一）

题　　目 语音信号的端点检测

专　　业 视听觉信息处理

学　　号 1180300419

班　　级 1803106 班

学　　生 刘晓慧

指 导 教 师 郑铁然

实 验 地 点 格物楼 207

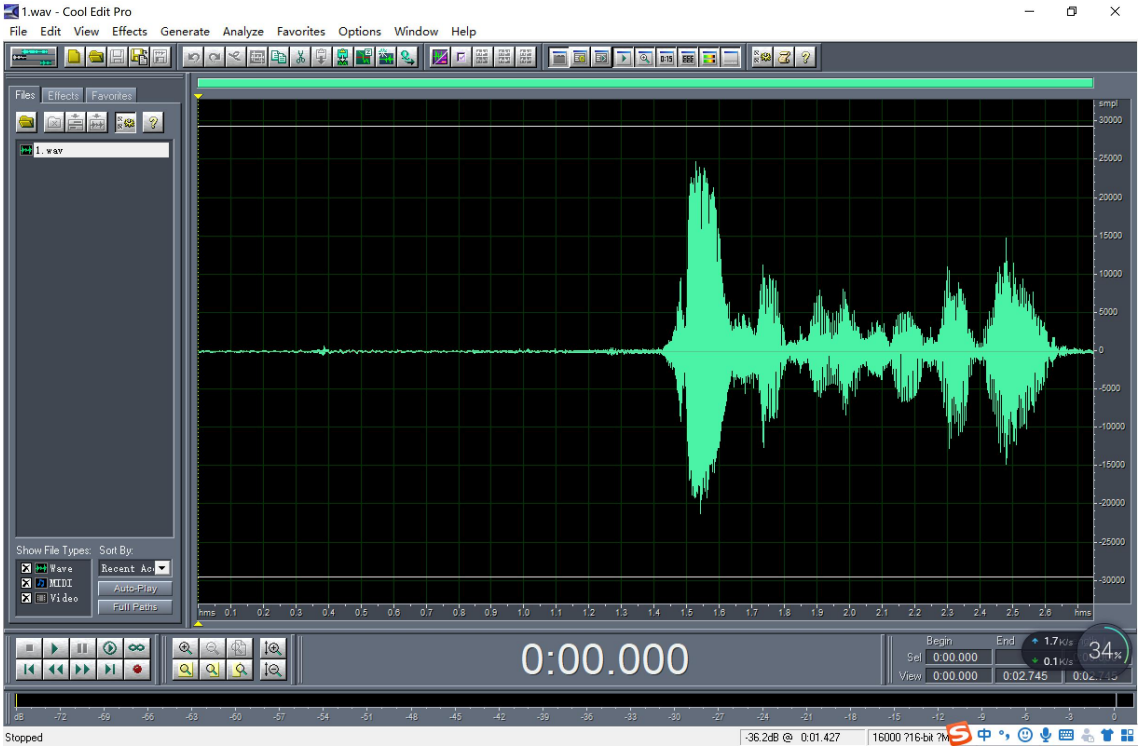
实 验 日 期 2020 年 12 月 16 日星期三

计算机科学与技术学院

一、 语音编辑和处理工具的使用

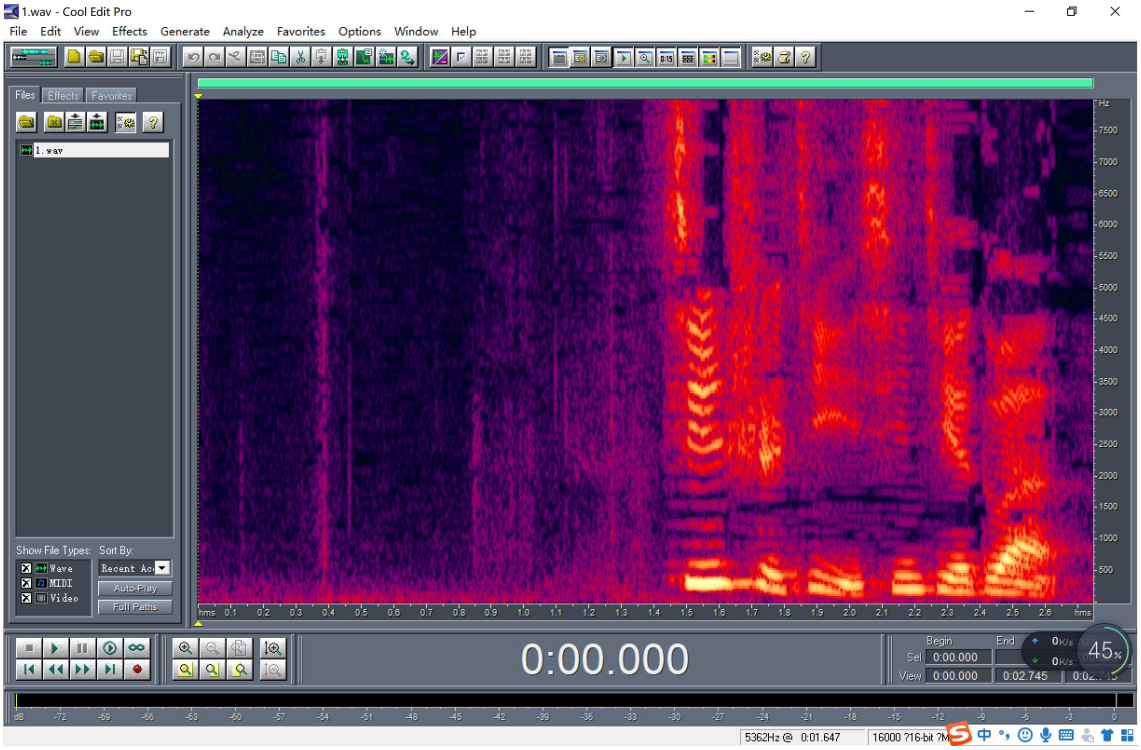
1.1 语音文件的时域波形截图

1.raw 在 Cool Edit Pro 中的波形显示如下图所示：

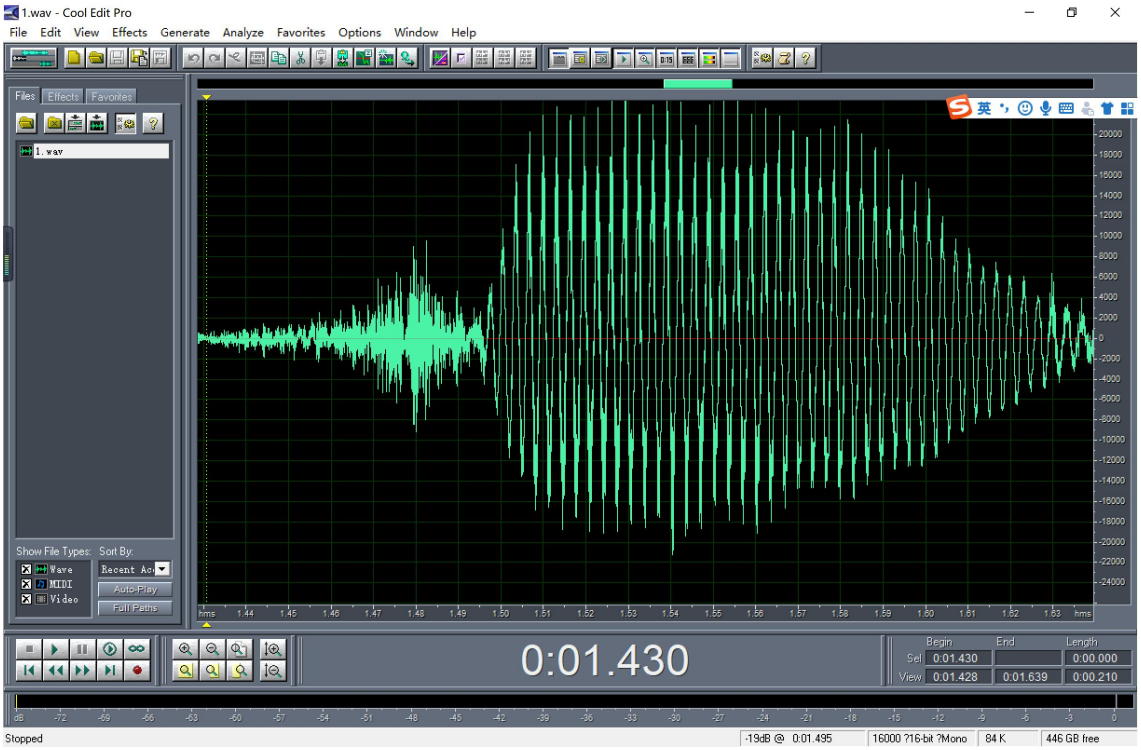


1.2 语音文件的语谱图截图

1.raw 的语谱图如下图所示：

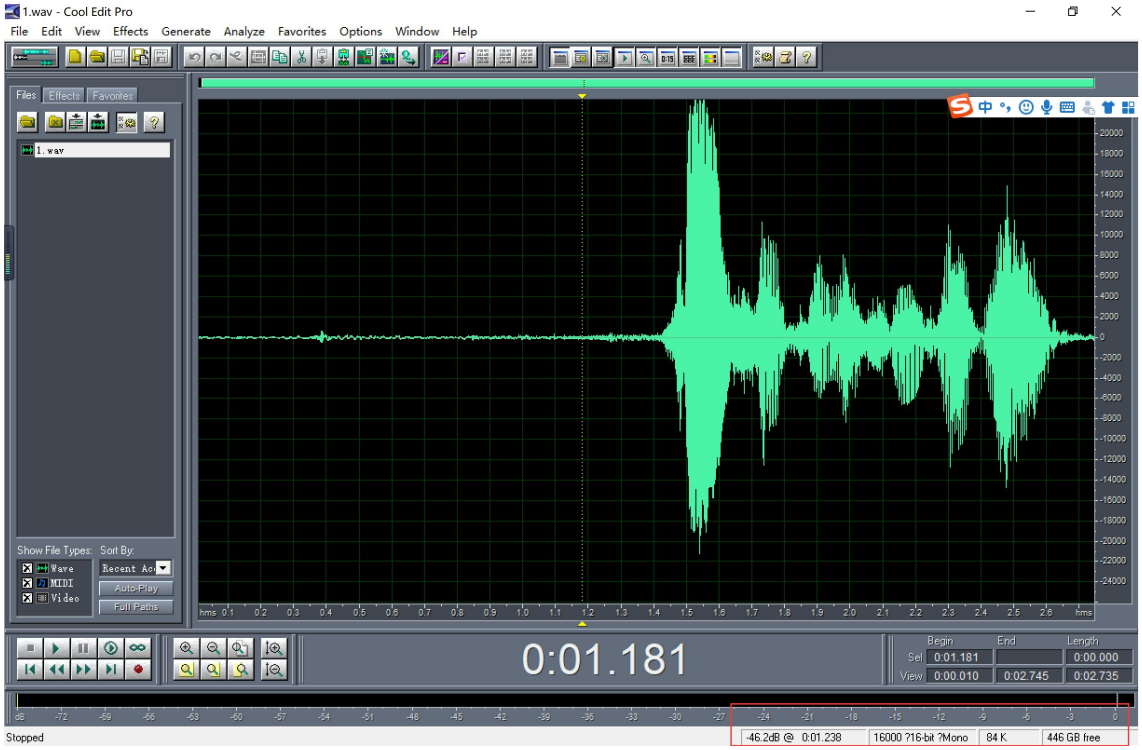


1.3 第一个音节的时域波形截图



1.4 语料的格式

采样频率　＝16K
量化比特数=16bit
声道个数　＝1 个，Mono 单声道



二、能量和过零率特征提取

2.1 给出特征提取算法，标明所采用的开发工具

开发工具：PyCharm Community Edition 2020.2.3 x64；Python3.6

1、计算各帧的能量的函数

短时能量的计算函数 $\mathrm{En}=\sum_{m=-\infty}^{\infty}[x(m)w(m)]^2$，且采取方窗进行计算，即 $\mathrm{w}(m)=1$。则利用以上公式，可以方便地得到下面的计算每一帧能量的函数：

```
# 计算传递进来的参数data的全部能量
def compute_powers(data):
    sum = 0
    for i in range(len(data)):
        sum += int(data[i]) * int(data[i])
    return sum
```

2、计算各帧的过零率的函数

短时过零率计算公式为 $\mathrm{Zn}=\sum_{m=-\infty}^{\infty}|\mathrm{sgn}[x(m)]-\mathrm{sgn}[x(m-1)]|\,w(m)$，其中 $w(m)$ 为 $1/2N$（N 为该帧的采样点个数），sgn 是符号函数。则由以上公式可以方便地得到下面的计算每一帧过零率的函数：

```
# 计算传递进来的参数data的过零率
def compute_zeros_rate(data):
    length = len(data)  # 获取字符串的长度
    signs = np.sign(data)  # 获取data的符号
    diff_value = np.abs(signs[1:] - signs[:length - 1])
    return np.sum(diff_value) / (2 * length)
```

3、预处理阶段的函数

在调用上面两个函数计算各帧的能量和过零率之前，首先需要读取 wav 文件，并且为了对声道数，采样频率，每个采样点所用帧数有个更加清晰的认识，将以上三个参数在控制台输出。利用 readframes（）获取采样点数据之后，很重要的一个操作是利用声道数和量化单位，将读取的二进制数据转换为一个可以计算的

数组。之后，便提取各帧的数据，传递到以上两个函数中，进行计算，将结果存储到相应文件中。具体代码如下：

```
for file_n in range(1, 11):
    # 打开wav文件，打开方式是读取二进制文件
    f = wave.open(r"./data/" + str(file_n) + ".wav", "rb")
    params = f.getparams()  # 获得文件的格式信息
    nchannels, sampwidth, framerate, nframes = params[:4]  # 依次获得各参数
    # 输出一些辅助参数
    print('该文件的声道数是: ', nchannels)
    print('该文件中样本点占用的字节数是: ', sampwidth)
    print('该文件的采样频率是: ', framerate)
    # 获得采样点的数据
    str_data  = f.readframes(nframes)
    f.close()
    #将波形数据转换成数组
    #需要根据声道数和量化单位，将读取的二进制数据转换为一个可以计算的数组
    wave_data = np.frombuffer(str_data,dtype = 'i2')
    frame_len = 256  # 设置帧长是256个采样点
    nframes = nframes // frame_len + 1  # 计算帧数
    # 用来存储各帧的能量值和过零率值
    powers = []
    zeros = []
    # 打开相应的文件，用来存储每帧的能量和过零率
    pow_file = open('./powers/'+ str(file_n) + '_en.txt', mode='w')
    zeros_file = open('./zeros/' + str(file_n) + '_zero.txt', mode='w')
    # 获取每一帧的数据，利用上面两个函数计算相应的值
    for i in range(nframes):
        if (i + 1) * frame_len < len(wave_data):
            frame_data = wave_data[i * frame_len : (i + 1) * frame_len]
        else:
            frame_data = wave_data[i * frame_len:]
        powers.append(compute_powers(frame_data))
        zeros.append(compute_zeros_rate(frame_data))
        pow_file.write(str(powers[i]) + '\n')
        zeros_file.write(str(zeros[i]) + '\n')
    pow_file.close()
    zeros_file.close()

    data_new = double_threshold_method(wave_data, powers, zeros, frame_len = 256)

    file_pcm = open('./pcm/' + str(file_n) + '.pcm', 'wb')
```

三、 端点检测算法

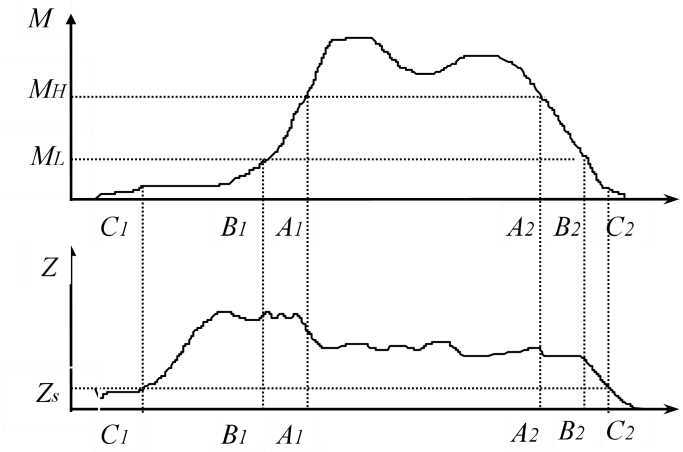
3.1 给出端点检测算法，标明所采用的开发工具

开发工具：PyCharm Community Edition 2020.2.3 x64；Python3.6

在 PPT 提供的两种检测方法中，我采取了双门限法来进行检测。

算法的关键是选取两类，三个阈值，分别是较高的短时能量门限 MH、较低的短时能量门限 ML、短时过零率 Zs，利用以上三个阈值进行端点检测。

算法的详细步骤如下：



1、首先利用较高的短时能量门限 MH 确保 A1-A2 部分肯定是浊音

2、从A1、A2开始向两侧搜索，短时能量值大于较低的短时能量门限ML的B1-B2段还是语音段

3、因为语音部分的过零率大于噪声部分，因此从 B1 开始向前搜索，短时过零率大于门限 Zs 的部分是清音部分

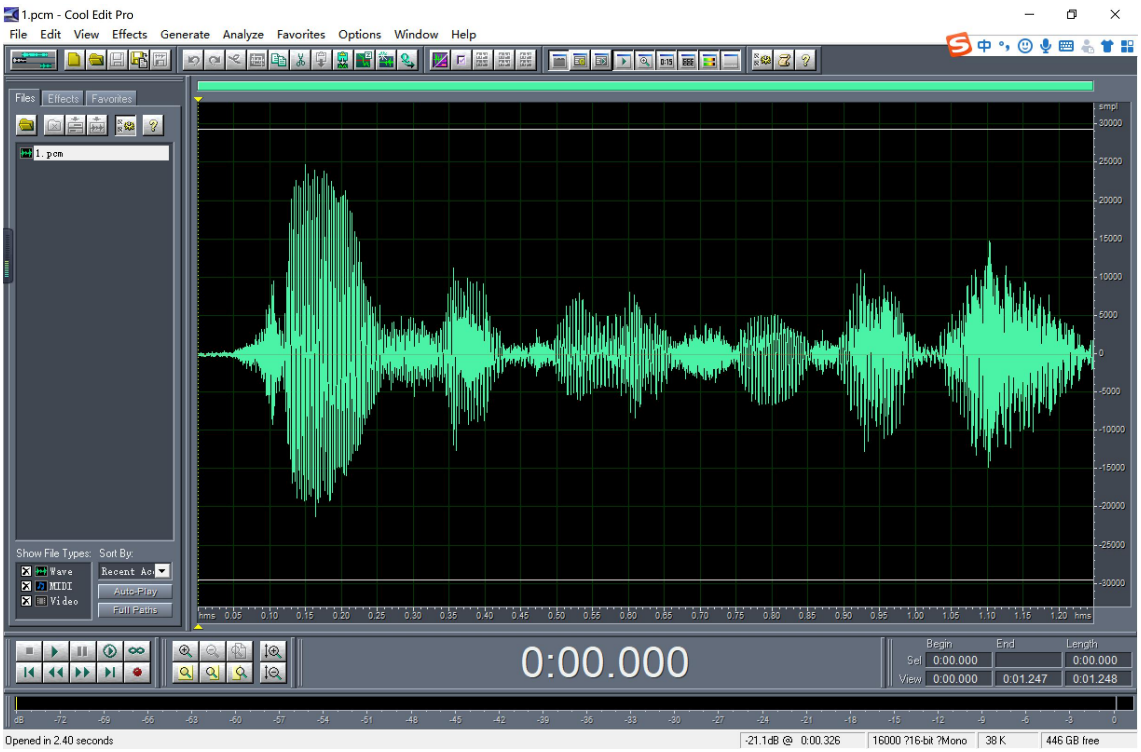
由以上步骤可以知道，三个阈值的选取很大程度上决定了实验的结果。根据本次实验所给的语料，有的语料噪音较多，有的语料噪音较小，因此采用自适应的方法，根据数据本身的特点选取三个阈值，且语料中，刚开始的部分都是噪声部分，因此前几帧的能量和过零率可以认为是非语音段的相应数据。多次实验之后查看最终效果，确定了下面的阈值：MH 设置为平均能量的四分之一，ML 设置为前四帧能量的平均值和 MH 的五分之一，Zs 设置为前四帧的平均值的三倍。

由以上分析得到下面的代码：

```
# 传递进的参数是采样数据，每一帧的能量数据 和 每一帧的过零率数据
def double_threshold_method(data, powers, zeros, frame_len):

    MH = np.average(powers) / 4  # 能量的高阈值
    ML = (np.average(powers[:4])+ MH) / 5  # 能量的低阈值
    ZS = np.average(zeros[:4]) * 3  # 过零率的阈值
    # 下面利用上面这三个、两类阈值 确定哪些是语音部分，哪些是噪声部分
    nframes = len(data) // frame_len + 1  # 因为要为每一帧划定标签，因此计算帧数为下面的处理做准备

    # 下面两个列表用来存储浊音部分和清音部分
    voiced_sound = np.zeros(nframes)
    unvoiced_sound = np.zeros(nframes)

    # 首先利用能量的高阈值确定浊音部分，同时将voiced_sound和unvoiced_sound相应的帧数标为1
    for i in range(nframes):
        if powers[i] > MH:
            voiced_sound[i] = 1
            unvoiced_sound[i] = 1
    # 利用能量的低阈值继续延伸语音段
    for i in range(len(voiced_sound)):
        # 因为最后一帧不是完整的，因此要判断是不是最后一帧
        if voiced_sound[i] == 1 and i > 0 and voiced_sound[i - 1] == 0:
            for j in range(i, 0, -1):
                if powers[j] > ML:
                    unvoiced_sound[j] = 1
                else:  # 一旦出现一个低于低阈值的段，则停止标记
                    break
        if voiced_sound[i] == 1 and i + 1 < nframes and voiced_sound[i + 1] == 0:
            for j in range(i + 1, nframes):
                if powers[j] > ML:
                    unvoiced_sound[j] = 1
                else:
                    break
    # 利用过零率值判断清音部分
    for i in range(len(unvoiced_sound)):
        # 因为最后一帧可能不是完整的，因此需要进行判断是不是最后一帧
        if unvoiced_sound[i] == 1 and i > 0 and unvoiced_sound[i - 1] == 0:
            for j in range(i - 1, 0, -1):
                if zeros[j] > ZS:
                    unvoiced_sound[j] = 1
                else:
                    break
        if unvoiced_sound[i] == 1 and i + 1 < nframes and unvoiced_sound[i + 1] == 0:
            for j in range(i, nframes):
                if zeros[j] > ZS:
                    unvoiced_sound[j] = 1
                else:
                    break
    return unvoiced_sound
```

四、 计算检测正确率

4.1 “1.wav”语料去除静音后的时域波形截图



4.2 正确率

正确检出文件的个数：10

正确率=　　100 %

五、 总结

10.1 请总结本次实验的收获

通过本次实验，第一次接触到了对于语音文件的处理。掌握了如何利用工具 Cool Edit Pro 辅助语音文件的处理，掌握了如何利用 Python 操作 wav 文件，并将课内的理论知识用于实战，自己实践了短时能量，过零率的计算以及双门限法的实现。并在一次次试错的过程中，加深了对理论知识的理解和认识。

10.2 请给出对本次实验内容的建议

实验难度合适，工作量比较小，可以顺利完成。可以适当丰富一些实验内容，增加一些项目。

六、附录文件解析

main.py 为代码文件
data 文件夹中为语料文件
pcm 文件夹中为进行端点检测后的语料文件
powers 文件夹中为各语料文件各帧的能量值
zeros 文件夹中为各语料文件各帧的过零率