

http://www.nit.net.cn/__local/C/EE/E5/AEF522EFB83938A8BD165923DEC_001BEC5E_9976.png

**信号与系统科技报告**



题 目 根据音频拨号音，识别出当前所拨号码

姓 名 王浩然、张文浩、张坚城

学 号 3190432060

专业班级 电信192

指导教师 王一刚

学 院 信息科学与工程学院

日 期 2020年12月28日

前言

音频拨号音又称双音多频（DTMF:Dual Tone Multi-frequency）信号，在世界范围内广泛应用在按键式电话机上的一种电信标准，近年来DTMF也应用在交互式控制中，诸如语音菜单、语音右键，电话银行和来电显示等功能上。

音频识别类似于语音识别，语音识别是一门交叉学科，语音识别正逐步成为信息技术中人机接口的关键技术。语音技术的应用已成为一个具有竞争性的新兴高技术产业。语音识别的研究工作可以追溯到20世纪50年代贝尔实验室的Audry系统，20世界80年代末，语音识别技术产生了巨大突破，其显著特征是HMM模型和人工神经元网络在语音识别中的成功应用。如今，我国语音识别技术正逐步稳定发展。本文就语音识别的一个部分进行研究。

关键词：双音多频；Python；快速傅里叶变换

# 基本原理与数学模型

经查阅相关资料，拨号音是一种双音多频信号DTMF（Dual Tone Multi Frequency），由高频群和低频群组成，高低频群各包含4个频率。一个高频信号和一个低频信号叠加组成一个组合信号，代表一个数字。DTMF信号有16个编码。

[双音多频](https://baike.baidu.com/item/%E5%8F%8C%E9%9F%B3%E5%A4%9A%E9%A2%91)的拨号键盘是4×4的[矩阵](https://baike.baidu.com/item/%E7%9F%A9%E9%98%B5" \t "_blank)，每一行代表一个低频，每一列代表一个高频。每按一个键就发送一个高频和低频的正弦信号组合，具体各数字对应的频率如表1-1所示。

表格 1‑1 理论DTMF频率及其对应的键值

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 低频群/Hz | 高频群/Hz | | | |
| 1209 | 1336 | 1477 | 1633 |
| 697 | 1 | 2 | 3 | A |
| 770 | 4 | 5 | 6 | B |
| 852 | 7 | 8 | 9 | C |
| 941 | \* | 0 | # | D |

例如，当在手机拨号盘上按下“5”时，则会产生频率为770Hz和1336Hz的两个正弦波信号，并叠加。DTMF信号规定，当传输每一个数字时，该信号必须持续大于等于45ms，且小于等于55ms。之外，100ms中其他的部分都无信号。

想要识别拨号音，必须先进行降噪，再进行检测识别。大多数情况下，使用硬件（如单片机）可以实现，软件方面对于连续按键声音的检测很少给出检测方案。本文通过Python软件编程来实现对DTMF信号的检测。

对DTMF信号的检测识别主要包括三个关键步骤：降噪，在输入的信号流中不断地搜索有输入音频对应的DTMF信号，将得到的DTMF信号分成高频和低频与已知频率群进行判断。

# 实施过程

2.1 样本制作与处理

首先需要制作样本，我们将手机扬声器放置在电脑的麦克风附近，在拨号盘上依次输入1、2、3、4、5、6、7、8、9、0、\*、#。电脑自带的录音软件的录制格式为m4a格式，如果直接处理可能较为麻烦，所以准备进行转码。经试验，大部分软件转码后的wav文件存在格式问题，并非原生的wav文件，在后续的音频分析识别中失败率较高。经查阅资料，最后决定使用python编写的ffmpeg进行转码，下列是单个音频文件转码前后的部分信息。

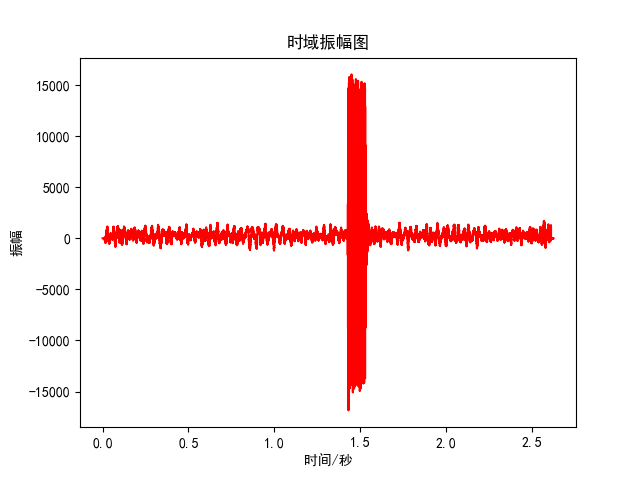
至此，我们可以使用Python自带的wave库进行处理。Wave库可以读取wav文件的通道数目，样本宽度，采样率和采样数目。了解库的用法后，我们导入数字0的样本，得到了时域振幅图（图2.1）：

图 2-1 降噪处理前时域振幅图

同样，我们使用Python中的numpy.fft进行傅里叶分析：

傅立叶原理表明：任何连续测量的时序或信号，都可以表示为不同频率的正弦波信号的无限叠加。

傅里叶变换即是把一条周期性曲线拆解成一组光滑正弦曲线的过程。

傅里叶变换的目的是可将时域（即时间域）上的信号转变为频域（即频率域）上的信号y=f(x)，随着域的不同， 对同一个事物的认知角度也随之改变，因此在时域中某些不好处理的地方，在频域就可以较为简单地处理。这就可以大量减少处理信号存储量。

假设有一时间域函数，根据傅里叶的理论它可以被分解为一系列正弦函数的叠加，他们的振幅A，频率ω或初相位φ不同：



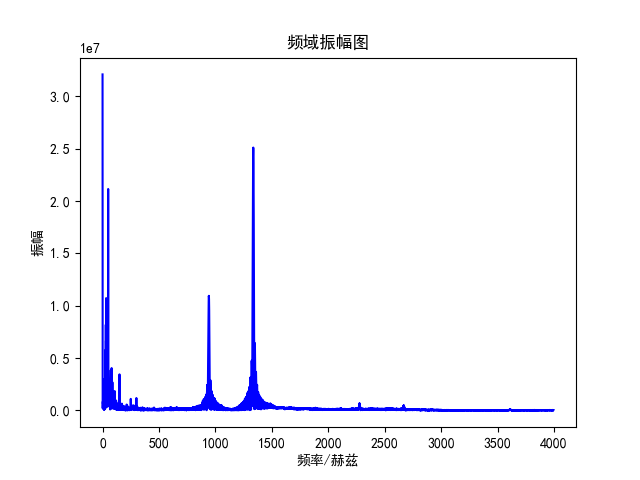
再去掉大于3000Hz以上的高频，得到频域振幅图（图2.2）

图 2-2 降噪处理前频域振幅图

从图中可以看出，直接进行分析的信号在低频段部分产生了许多毛刺，这是意料之中的，因此接下来就需要对音频信号进行降噪处理。

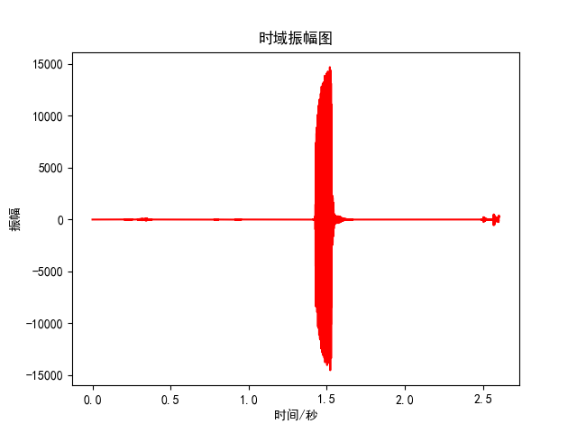
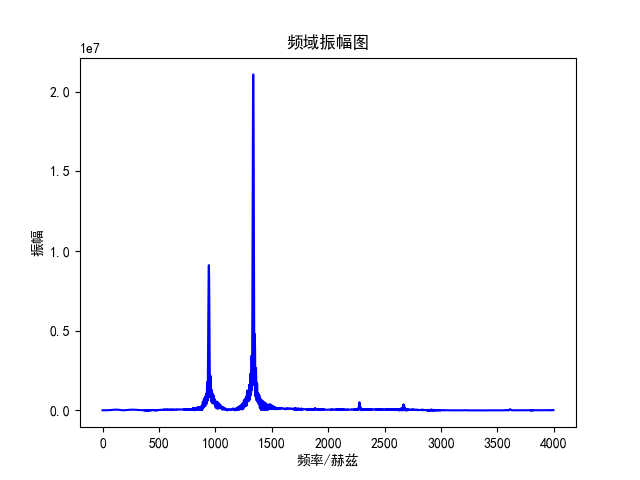
取一段噪声信号，命名为noise.wav，使用python的sox进行噪音采样，生成noise.prof，再对目标音频文件进行降噪处理，处理后的音频的时域和频域振幅分析如下：

图 2-3 降噪处理后时域振幅图

图 2-4 降噪处理后频域振幅图

# 与之前的分析结果对比，结果还是很理想的，在频域图中我们能清晰的看出数字“0”是有一个低频信号和一个高频信号叠加而成的，符合DTMF信号的定义。以此类推，我们做出了从0到9包括\*和#所有音频信号的频率点图（图2.5），其中水平坐标轴0到9分别对应数字0到9，\*和#分别为10和11。

从图2.5中我们能得出实际样本的双频（表2.1）。

表格 2‑1 实际样本的DTMF频率及其对应的键值

|  |  |  |  |
| --- | --- | --- | --- |
| 低频群/Hz | 高频群/Hz | | |
| 1210 | 1338 | 1479 |
| 700 | 1 | 2 | 3 |
| 773 | 4 | 5 | 6 |
| 855 | 7 | 8 | 9 |
| 945 | \* | 0 | # |

与表1-1比较，误差范围在1-3Hz左右，较符合理论结果，可继续进一步分析。

* 1. DTMF双音多频信号的识别

我们对样本进行了快速傅里叶变换来寻找特定的频率，frequencies包含一个从-4000Hz到4000Hz的326个频率的列表，amplitudes包含这326个频率的相对振幅（顺序相同）

1. frequencies = np.fft.fftfreq(signal.size, d=1/fps)
2. amplitudes = np.fft.fft(signal)

以低频部分举例，i\_min是频率大于0 Hz的一个变量，i\_max是频率大于1050hz的变量。这个函数就可以判断测试样本中的值是否与训练样本的低频率的值接近。

1. # Low
2. i\_min = np.where(frequencies > 0)[0][0]
3. i\_max = np.where(frequencies > 1050)[0][0]

# 接着设置如下变量：

1. freq = frequencies[i\_min:i\_max]
2. amp = abs(amplitudes.real[i\_min:i\_max])

# 在时域对信号进行裁剪，我们将这个范围内振幅较大的频率存储到lf中。之后，我们搜索DTMF的四个频率中哪一个最接近lf。

1. lf = freq[np.where(amp == max(amp))[0][0]]
2. delta = args.t
3. **for** f **in** [697, 770, 852, 941]:
4. **if** abs(lf-f) < delta:
5. delta = abs(lf-f)
6. best = f
7. lf = best

# 实验结果分析

# 设计好算法后，我们导入全键样本文件进行编译，得到如下结果：

1. python detector.py num\_all.wav
2. OutPut:123456789\*0#

完美的进行了识别，为了防止幸存者偏差，我们又临时录制了一段音频，经过同样的转码降噪处理后，得到了如下结果：

1. python detector.py 1145141919810\_10\_11.wav
2. OutPut:1144555141919810\*##

目标结果应该是1145141919810\*#，但是实际结果出现了相同数字连续出现的情况。对于这一结果，我们认为是算法的扫描精度过高（默认是0.1s），于是将每次的判断间隔稍微拉长，再次进行识别，得到如下结果：

1. python detector.py 1145141919810\_10\_11.wav
2. OutPut:1145141919810\*#

得到了理想结果。

# 结论

对于DTMF双音多频信号的分析，主要是要抓住信号的低频段和高频段，经过简单的傅里叶变换便能得出其规律。根据实际结果调整DTMF的双频范围，再逐段分割信号，与双频群进行比较判断便能得出结果。

实际上，该算法还有很多的不足，主要的缺点是识别器对样本文件的降噪要求较高，在多次试验中结果的误差较大，Sox有时不能满足识别器的降噪要求，便无法得出理想的结果。