DTMF 信号的检测与识别

无 76　RainEggplant　2017******

程序代码

第一题

环境：

- mingw-w64
- C++ 17

文件清单（有使用开源代码处理音频读写和寻峰，详见 `README.md`）：

- p1.exe
- p1.cpp
- AudioFile.h
- dsp/
  - fft.h, fft.cpp
  - utils.h, utils.cpp

为了在第三题用 fft，最后又写了 matlab 版，所以还有：

- p1.m
- get_key_fft.m

第二、三题

（老师后来说可以用 MATLAB 了，所以就上的 MATLAB 了，但是 `goertzel` 函数仍然是自己实现的）

文件清单：

- p2.m
- p3.m
- my_goertzel.m

设计思路

第一题

- 调用上次作业中编写的 FFT 函数，对音频执行 DFT，获得音频的频谱。
- 在 $[0, \frac{1700}{fs}\pi]$ 频率区间寻找最大的两个峰，并根据采样率转换为模拟频率。
- 根据两个模拟频率查表，得出按键。

注：在 Matlab 版本的基于 FFT 的方法里，`findpeaks` 寻峰的条件更加精确，采用了如下参数：

- Threshold = max(Y_abs) * 0.01
- MinPeakDistance = floor(200 / fs * N)：即利用所求峰值有最小间距的先验知识。

`my_goertzel` 函数

- 接受音频序列和采样率为参数
- 根据采样率计算 8 个感兴趣的频率所对应的 k 值
- 分别对这 8 个 k, 迭代地计算差分方程 $v_k[n] = 2\cos(\omega_k)v_k[n-1] - v_k[n-2] + x[n]$
- 计算 $X[k] = v_k[N] - W_N^k v_k[N-1]$
- 寻找最大的两个 $X[k]$
- 查表转换为按键

第二题

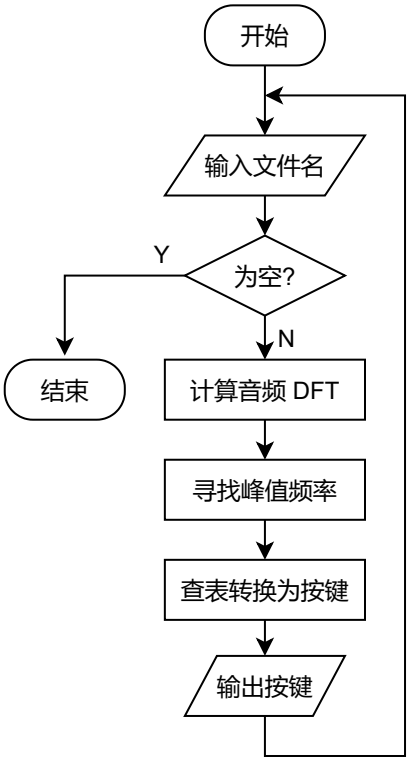
- 调用 `my_goertzel` 获取按键即可

第三题

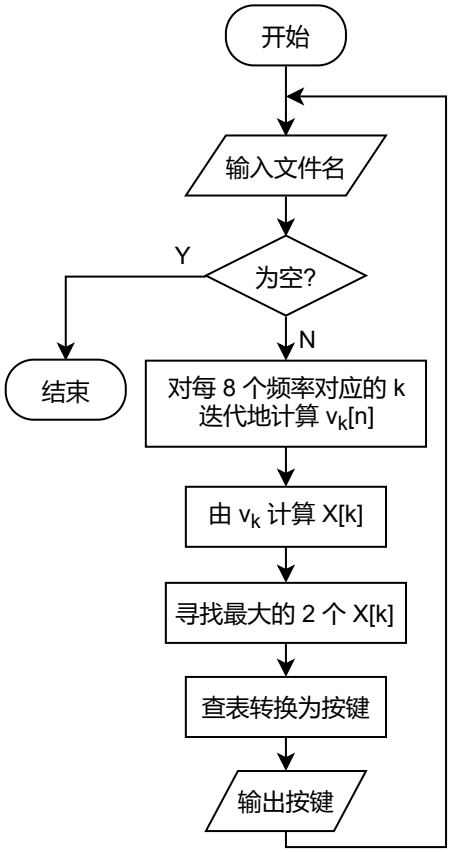
- 对音频分帧，计算 RMS
- 根据 RMS 分出有效的音频片段
- 对每个片段，调用 `get_fft_key` 或 `my_goertzel` 获取按键

流程图

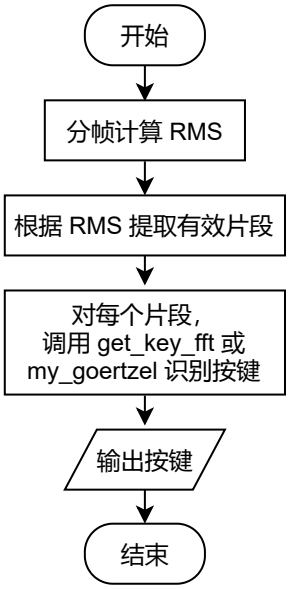
第一题



第二题


开始
输入文件名
为空?
Y
N
结束
对每 8 个频率对应的 k
迭代地计算 $v_k[n]$
由 v_k 计算 X[k]
寻找最大的 2 个 X[k]
查表转换为按键
输出按键

第三题


开始
分帧计算 RMS
根据 RMS 提取有效片段
对每个片段，
调用 get_key_fft 或
my_goertzel 识别按键
输出按键
结束

计算结果

第一题

```
Input filename (empty line to exit): data1/data1081.wav
Key: 5
Input filename (empty line to exit): data1/data1107.wav
Key: 1
Input filename (empty line to exit): data1/data1140.wav
Key: 6
Input filename (empty line to exit): data1/data1219.wav
Key: 9
Input filename (empty line to exit): data1/data1234.wav
Key: 8
Input filename (empty line to exit): data1/data1489.wav
Key: 7
Input filename (empty line to exit): data1/data1507.wav
Key: 3
Input filename (empty line to exit): data1/data1611.wav
Key: 4
Input filename (empty line to exit): data1/data1942.wav
Key: 0
Input filename (empty line to exit): data1/data1944.wav
Key: 2
```

第二题

```
>> p2
Input filename (empty line to exit): 'data1/data1081.wav'
Key: 5
Input filename (empty line to exit): 'data1/data1107.wav'
Key: 1
Input filename (empty line to exit): 'data1/data1140.wav'
Key: 6
Input filename (empty line to exit): 'data1/data1219.wav'
Key: 9
Input filename (empty line to exit): 'data1/data1234.wav'
Key: 8
Input filename (empty line to exit): 'data1/data1489.wav'
Key: 7
Input filename (empty line to exit): 'data1/data1507.wav'
Key: 3
Input filename (empty line to exit): 'data1/data1611.wav'
Key: 4
Input filename (empty line to exit): 'data1/data1942.wav'
Key: 0
Input filename (empty line to exit): 'data1/data1944.wav'
Key: 2
```

可见二者的计算结果一致。

第三题

使用 FFT

```
>> p3
Enter "0" to use FFT-based method: 0
Keys: 2058911320X1649
```

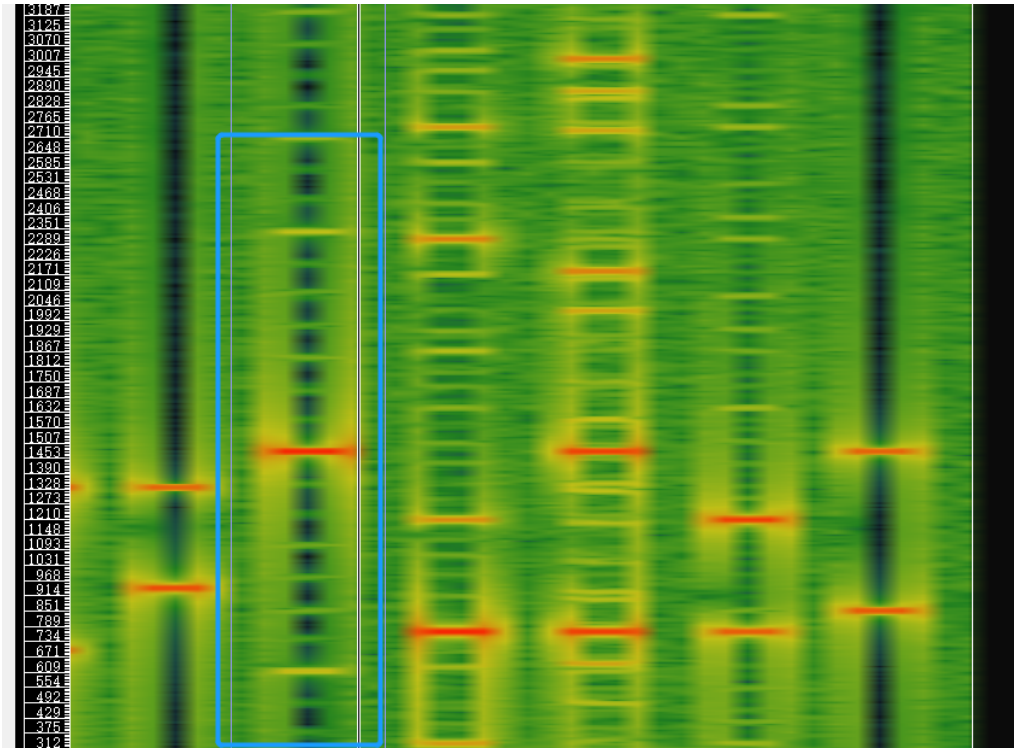
'X' 表示识别失败。

使用 Goertzel

```
>> p3
Enter "0" to use FFT-based method: 1
Keys: 205891132064649
```

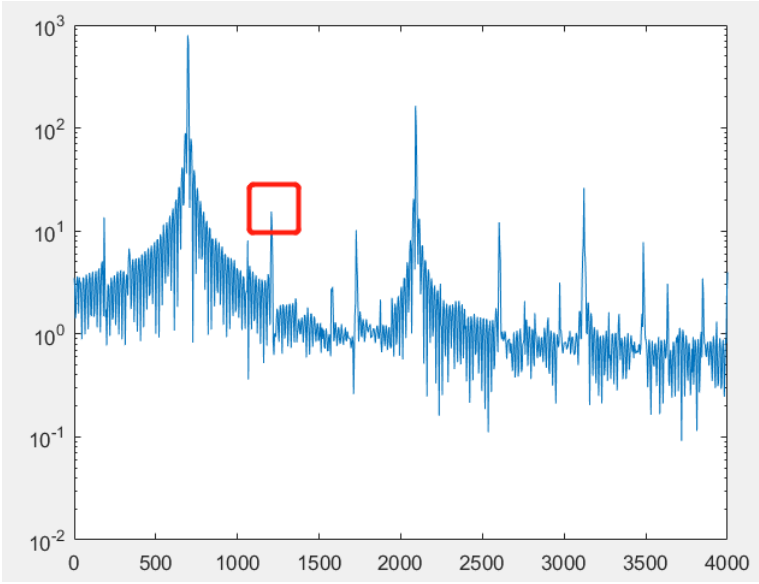
除去倒数第 5 处，两种方法的结果一致。仔细检查发现，本题的数据可能 **存在错误** 。体现在第 7 处和倒数第 5 处。

倒数第 5 处时频图如下：

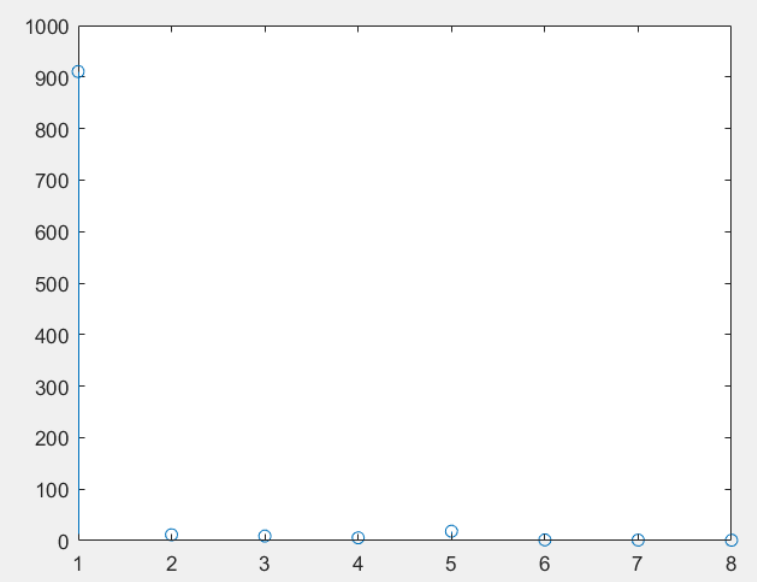


可见仅存在 1477 Hz 的有效频率，下面那根大概是 610 多 Hz，不为有效频率。

第 7 处频谱如下（纵轴取对数，否则第二个频率看不出来）：



Goertzel 谱如下：



的确有存在两个有效频率，但第二个频率的幅度非常低，很容易发生误判。不过这里两种算法还是经受住了考验。

复杂度比较

基于 FFT 的识别算法

采用基 2 FFT 算法计算 DFT，对一个长为 n 的片段（忽略补零），时间复杂度为 $\mathcal{O}(N\log N)$。具体来说，大致需要 $1/2N\log_2 N$ 次复乘和 $N\log_2 N$ 次复加。

执行 DFT 后需要进行寻峰，该部分的时间复杂度为 $\mathcal{O}(N)$。

总的时间复杂度为 $\mathcal{O}(N\log N)$。

基于 Goertzel 的识别算法

采用 Goertzel 算法，对一个长为 n 的片段，由于只关心 8 个频点，故大致只需要 $2N\times 8=16N$ 次实乘、$2N\times 8=16N$ 次实加和 8 次复乘、8 次复加。故该部分的时间复杂度为 $\mathcal{O}(N)$。

执行 Goertzel 后需要寻找最大的两个值，由于总共频点为 8 个，该部分为常数时间复杂度。

总的时间复杂度为 $\mathcal{O}(N)$。

可见，时间复杂度上，基于 Goertzel 的识别算法是要优于基于 FFT 的识别算法的。