信号处理原理实验

张晨

2020 年 1 月 7 日

1 双音频按键识别

1.1 实验原理

电话拨号时，按下每个键会发出不同的声音，这是因为每个按键的声音包括两个不同频率的正弦信号。将时域上的声音信号转变到频域上，即可提取出这两个正弦信号，从而识别出按下的按键。

将时域信号变为频域信号的常见做法有以下两个：

1. FFT，以 $O(n \log n)$ 的时间复杂度求出各个频率的分量大小。
2. Goertzel，以 $O(n)$ 的时间复杂度求出某一个频率的分量大小。

实验说明中给出了 Goertazl 算法的原理式。

$$\begin{aligned} v_k(n) &= x(n) + 2\cos\left(\frac{2\pi k}{N}\right) v_k(n-1) - v_k(n-2) \\ y_k(n) &= v_k(n) - v_k(n-1)W_N^k \\ v_k(-1) &= v_k(-2) = 0 \end{aligned} \tag{1}$$

其中，第二个式子可进一步化简为

$$y_k(n) * y_k^*(n) = v_k(n-1)^2 + v_k(n)^2 - 2\cos\left(\frac{2\pi k}{N}\right) v_k(n-1)v_k(n) \tag{2}$$

具体实现时，将音频信号切分成若干小段，分别在频域上进行分析提取出按键，最后再进行去重。

1.2 实验结果

我使用了两个测例进行测试。一个是录下的 $123456789 * 0\#$，下称手机音频，另一个是从 B 站获取的使用按键演奏欢乐颂的片段，下称欢乐颂。程序输出如图1所示。

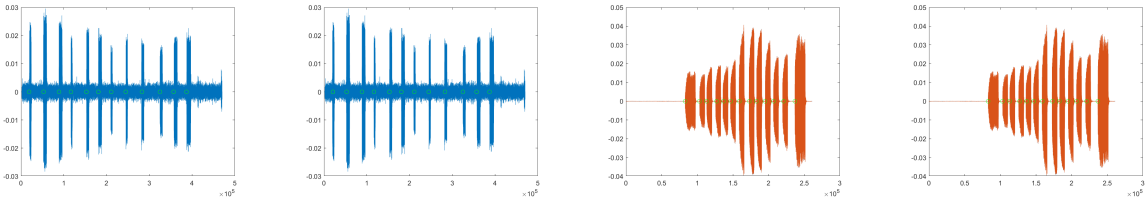
1.2.1 准确性

这两个算法均能准确识别出按键音，但是欢乐颂中存在多次快速按下同一个键的情况，暂未找到分开这些按键音的方法。识别的效果如图2所示。

```
Testcase: test
时间已过 0.157199 秒。
fft output: 123456789*0#
时间已过 0.075934 秒。
goertzel output: 123456789*0#

Testcase: joy
时间已过 0.091859 秒。
fft output: 3699632112332
时间已过 0.049691 秒。
goertzel output: 3699632112332
```

图 1: 实验 1 程序输出



(a) 手机音频，FFT (b) 手机音频，Goertzel (c) 欢乐颂，FFT (d) 欢乐颂，Goertzel

图 2: 按键识别情况，识别出的按键以圆圈标出。

1.2.2 运行效率

两算法的效率对比如表1所示。可以看出，Goertzel 的速度大致是 FFT 的两倍。

	FFT	Goertzel
手机音频	0.157199 秒	0.075934 秒
欢乐颂	0.091859 秒	0.049691 秒

表 1: 两种算法的运行效率对比

2 卷积计算方法的性能比较

2.1 实验原理

总计实现了以下 4 种卷积计算方法

1. 直接计算法直接套用卷积的公式
2. 圆卷积法 $y(n) = x(n) * h(n) = IDFT[DFT(x) \cdot DFT(h)]$，为使用快速傅里叶变换加速，应将序列补齐至长度为 2 的整数次幂。
3. Overlap Add 将序列 x 拆分成若干段，每段分别计算与 $h(n)$ 的卷积，最后再累加起来。

4. Overlap Save 将结果序列拆分成若干段，分别进行计算。

2.2 实验结果

Overlap Add 和 Overlap Save 的运行时间与切分长度有密切的关系。通过实验，我发现在 Overlap Add 中，切分长度取 3 倍 $h(n)$ 的长度左右比较合适，Overlap Save 则需要取到 9 倍 $h(n)$ 的长度左右。以下实验都基于这组参数。

2.2.1 直接计算法与其他方法

实验结果如图3所示。可以发现，直接计算法 (bruteforce) 明显慢于其他的做法。这是因为其时间复杂度为 $O(n^2)$，而其他做法的复杂度都为 $O(nlogn)$。

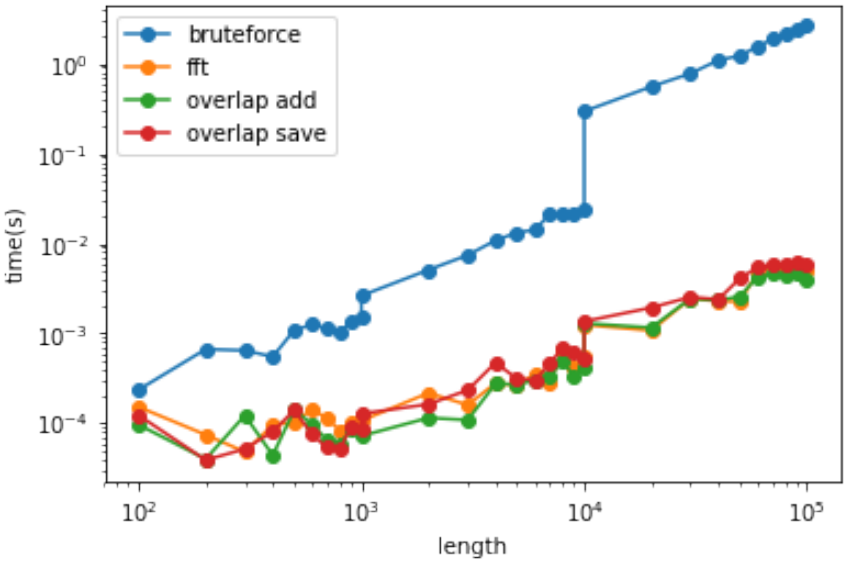
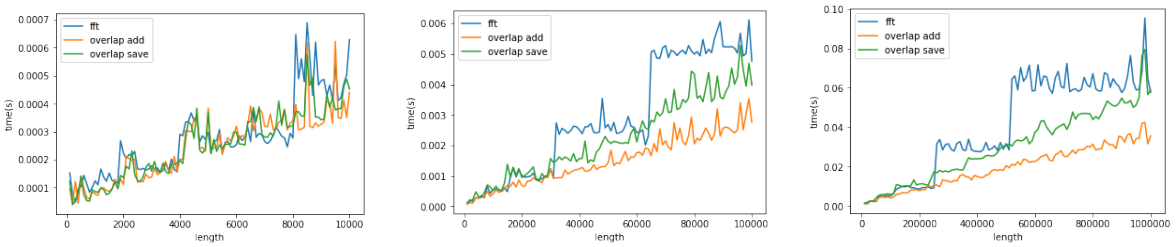


图 3: 四种方法的运行时间比较。横坐标为 x 的长度。x 长 100 至 900 时，h 长 100；x 长 1000-9000 时，h 长 1000；x 长 10000 至 90000 时，h 长 10000。坐标皆为对数坐标。

2.2.2 Overlap 的效果

从图4中可以发现，只有在两序列长度相差悬殊的时候，overlap 才会起到作用。Overlap Add 的性能要好于 Overlap Save，这是因为 Overlap Add 引入的冗余计算仅仅为若干次加法运算，而 Overlap Save 却引入了更多的傅里叶变换的计算。



(a) len(h(n))=100　　(b) len(h(n))=1000　　(c) len(h(n))=10000

图 4: 两种 Overlap 方法与圆卷积法的运行时间对比，横坐标为 x(n) 的长度。

3 语音信号的频分复用

3.1 实验原理

语音主要分布在一个很狭小的频带里，因此可以将多个语音信号在频域上依次排开，达到频分复用的效果。

3.2 实验结果

我从新闻联播上录了三段音频，它们的原始时域信号如图5所示。

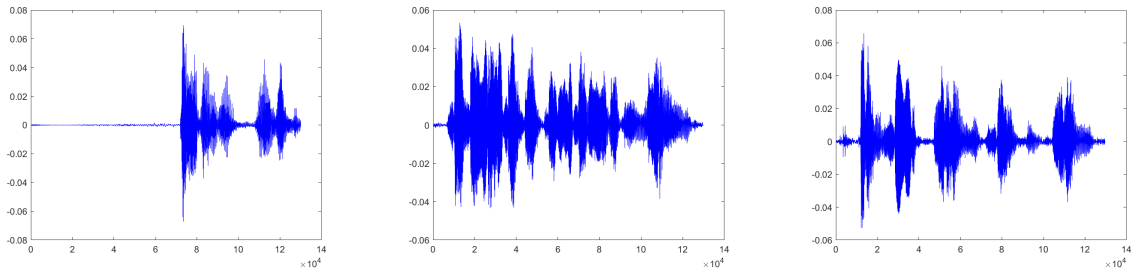


图 5: 原始音频的时域信号

3.2.1 调制过程

分别将它们进行傅里叶变换，得到的频域信号如图6所示。

之后，截取每个频域信号的前 20000 个点和后 20000 个点，在频域上依次排布，得到合成后的频域信号，如图7(a)所示。将得到的频域信号逆变换回时域，便得到合成后的音频。

3.2.2 解调过程

读入合成后的音频，进行傅里叶变换，恢复其频域信号，如图7(b)所示。将所得的频域信号拆分成 3 个音频所对饮的频域信号，如图8所示。

之后，对这三个频域信号分别进行傅里叶逆变换，即可得到恢复的时域信号，如图9所示。对比图5可以发现，恢复后的音频几乎没有丢失内容。

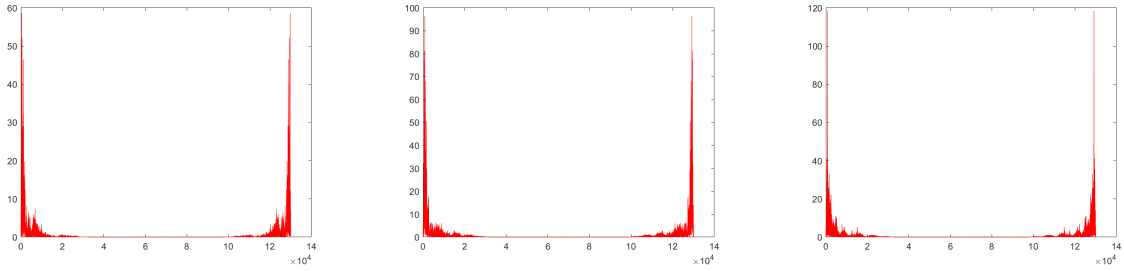
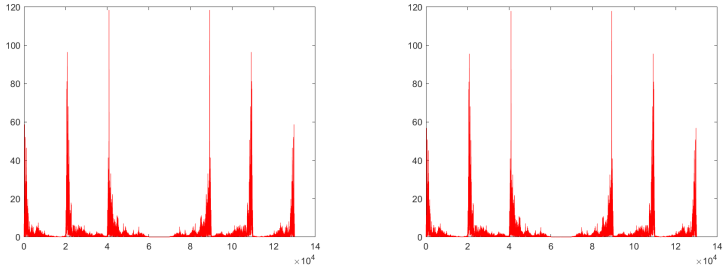


图 6: 原始音频的频域信号



(a) 合成后的频域信号　　(b) 合成音频恢复的频域信号

图 7: 合成后的频域信号与从合成音频恢复的频域信号

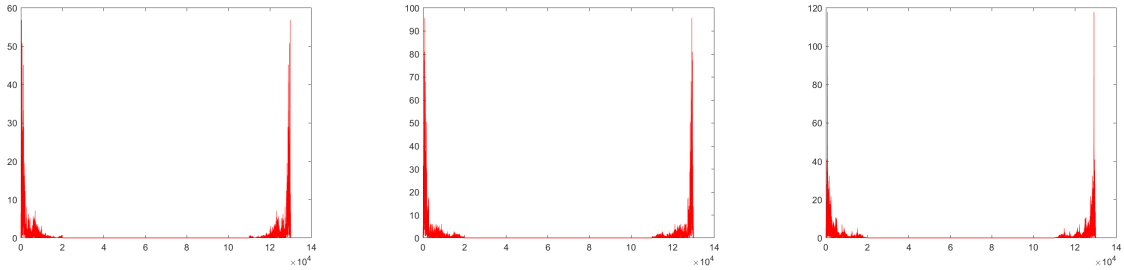


图 8: 合成音频拆分后的频域信号

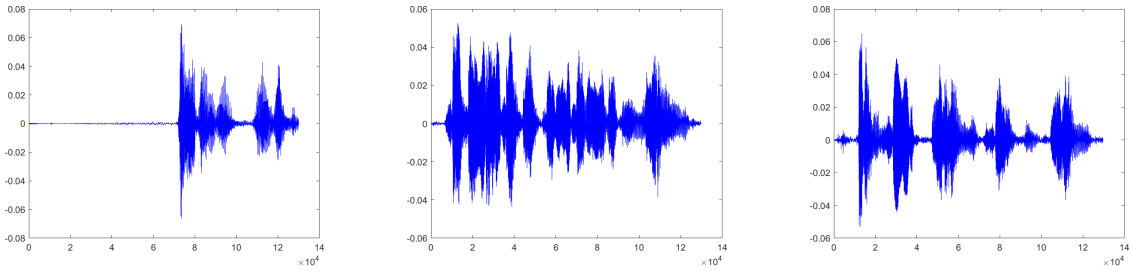


图 9: 恢复的时域信号