








公告

Since 2017.05.21:

Visitors

 CN 164,792	 GB 1,255
 US 6,915	 DE 977
 TW 3,449	 SG 833
 HK 2,890	 AU 827
 JP 1,844	 CA 650

Pageviews: 307,480

FLAG counter

昵称: 桂。

园龄: 1年8个月

粉丝: 260

关注: 15

+加关注

<	2018年9月											>
日	一	二	三	四	五	六						
26	27	28	29	30	31	1						
2	3	4	5	6	7	8						
9	10	11	12	13	14	15						
16	17	18	19	20	21	22						
23	24	25	26	27	28	29						
30	1	2	3	4	5	6						

最新随笔

1. Mathematica简介
2. 概率论02
3. 概率论01
4. 小数延迟滤波器的诸多工程问题及改进措施
5. coon's patch
6. system generator学习笔记【02】
7. CZT变换 (chirp z-transform)
8. system generator学习笔记【01】
9. 硬件资源拆解
10. 基础008_定浮点转化[floating point IP]

随笔分类

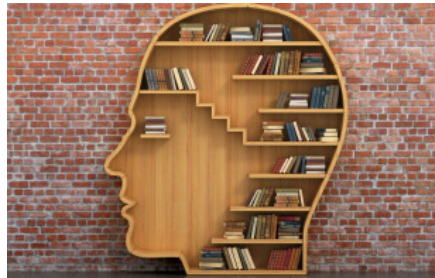
- 00-C 语言(2)
- 01-MATLAB(12)
- 02-Python(9)
- 03-前端相关(1)
- 10-HTK(1)
- 11-Linux(3)
- 12-Tensorflow(2)
- 13-scikit-learn(4)
- 14-工具使用(16)
- 15-音频分析工具(2)
- 16-硬件(39)
- 20-音频信号(30)
- 21-信号处理(55)
- 22-图像(3)
- 23-模式识别(18)
- 24-矩阵(18)
- 25-测向算法(32)
- 26-电磁(2)
- 27-数学(2)
- 待归档(4)

随笔-239 文章-6 评论-174

音频特征提取——常用音频特征

作者: 桂。

时间: 2017-05-05 21:45:07

链接: <http://www.cnblogs.com/xingshansi/p/6815217.html>

前言

主要总结一下常用的音频特征, 并给出具体的理论分析及代码。

一、过零率

过零率的表达式为:

$$Z_n = \frac{1}{2} \sum_{m=0}^{N-1} | \text{sgn}[x_n(m)] - \text{sgn}[x_n(m-1)] |$$

其中, $\text{sgn}[\]$ 是符号函数, 即:

$$\text{sgn}[x] = \begin{cases} 1, & (x \geq 0) \\ -1, & (x < 0) \end{cases}$$

其中 N 为一帧的长度, n 为对应的帧数, 按帧处理。

理论分析: 过零率体现的是信号过零点的次数, 体现的是频率特性。因为需要过零点, 所以信号处理之前需要中心化处理。

code(zcr1即为过零率):

```
1 for i=1:fn
2     z=X(:,i); % 取得一帧数据
3     for j=1:(wlen-1); % 在一帧内寻找过零点
4         if z(j)*z(j+1)<0 % 判断是否为过零点
5             zcr1(i)=zcr1(i)+1; % 是过零点, 记录1次
6         end
7     end
8 end
```

二、短时能量

短时能量的表达式为:

设第 n 帧语音信号 $x_n(m)$ 的短时能量用 E_n 表示, 则其计算公式为:

$$E_n = \sum_{m=0}^{N-1} x_n^2(m)$$

式中, N 为信号帧长。

理论分析: 短时能量体现的是信号在不同时刻的强弱程度。

code:

读书(26)
工作记录(12)
随手记(7)

随笔档案

2018年6月 (4)
2018年5月 (20)
2018年4月 (4)
2018年3月 (2)
2018年2月 (11)
2018年1月 (4)
2017年12月 (3)
2017年11月 (11)
2017年10月 (13)
2017年9月 (14)
2017年8月 (18)
2017年7月 (14)
2017年6月 (16)
2017年5月 (37)
2017年4月 (29)
2017年3月 (25)
2017年2月 (10)
2017年1月 (4)

积分与排名

积分 - 165677
排名 - 1841

最新评论

1. Re:MATLAB (2) ——小波工具箱使用简介

@若知, 问题解决了吗, 求指点一二

--大枫侠
2. Re:信号处理——EMD、VMD的一点小思考

你好, 博主, 我做的轴承故障, 帮忙看下VMD选择几层分解, 根据图。

--641573672
3. Re:信号处理——EMD、VMD的一点小思考

博主你好, 请问在VMD中 (function [u, u_hat, omega] = VMD(signal, alpha, tau, K, DC, init, tol)) , 每次K值下会求得相应的中心频率.....

--蓝色砂砾
4. Re:PCA算法

真的很详细,

--GXTon
5. Re:稀疏傅里叶变换 (sparse FFT)

@陈曦明你看的那个论文, P25, sigma^2 = B^2*log2, sigma与M有数学上的联系, 自己可以换算一下, 细节就不帮你看了。...

--桂。

阅读排行榜

1. python音频处理用到的操作(25774)
2. 音频特征提取——librosa工具包使用(18920)
3. 信号处理——Hilbert变换及谱分析(15995)
4. PCA算法(12337)

```
1 for i=1 : fn
2     u=X(:,i);           % 取出一帧
3     u2=u.*u;            % 求出能量
4     En(i)=sum(u2);      % 对一帧累加求和
5 end
```

三、短时自相关函数

短时自相关函数定义式为：

$$R_n(k) = \sum_{m=0}^{N-1-k} [x(n+m)w'(m)][x(n+m+k)w'(k+m)]$$

理论分析：学过信号处理的都应该知道，信号A与信号B翻转的卷积，就是二者的相关函数。其中 $w'(m)$ 是因为分帧的时候，加了窗函数截断，w代表窗函数。

code:假设一帧截断的信号

```
1 r = xcorr(signal);
```

这与直接利用卷积的方式等价：

给出卷积的实现：

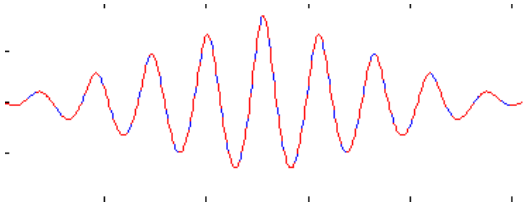
```
1 function output_signal=my_direct_convolution(input_signal,impulse_response)
2 % Input:
3 %     input_signal: the input signal
4 %     impulse_response: the impulse response
5 % Output:
6 %     output_signal:the convolution result
7 N=length(input_signal);%define length of signal
8 K=length(impulse_response);%define length of impulse response
9 output_signal=zeros(N+K-1,1);%initializing the output vector
10 xp=[zeros(K-1,1);input_signal;zeros(K-1,1)];
11 for i=1:N+K-1
12     output_signal(i)=xp(i+K-1:-1:i) '*impulse_response;
13 end
```

卷积也可以借助FFT快速实现。

调用卷积的函数：

```
1 r1 = my_direct_convolution(signal,signal(end:-1:1));
```

图中可以看出r与r1完全等价：



四、短时平均幅度差

假设x是加窗截断后的信号，短时平均幅度差定义：

$$r_n(k) = \sum_0^{N-1} |x(n) - x(n+k)|, k = 0,1, \dots, N-1$$

理论分析：音频具有周期特性，平稳噪声情况下利用短时平均幅度差可以更好地观察周期特性。

code:

取一帧信号，计算短时平均幅度差：

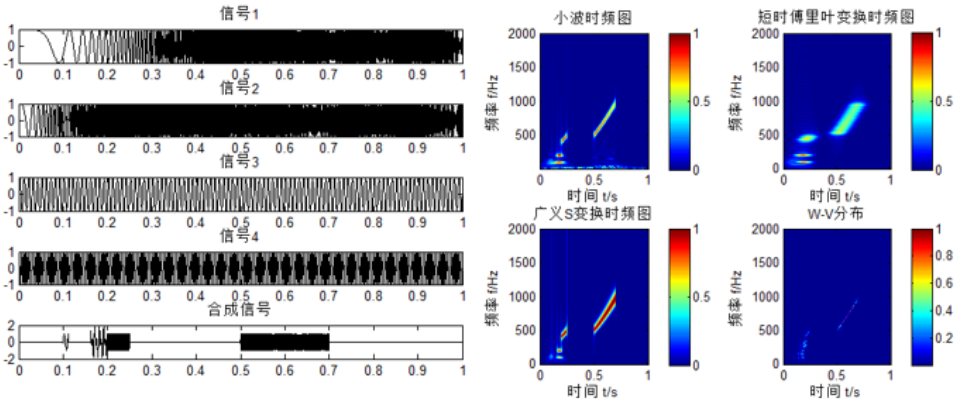
```
1 u = X(:,i) %取一帧信号
2 for k = 1:wlen
3     amdvec(k) = sum(abs(u(k:end)-u(1:end-k+1))) ; %求每个样点的幅度差再累加
4 end
```

前面四个都是信号的时域分析，音频信号更多是在时频域分析(可借助tftb-0.2工具包分析)。

推荐排行榜

- 1. 信号处理——EMD、VMD的一点小思考(9)
- 2. 音频特征提取——常用音频特征(7)
- 3. python音频处理用到的操作(6)
- 4. 霍夫变换(4)
- 5. 统计学习方法：感知机(3)

常用的有STFT（短时傅里叶变换）、小波变换、ST、W-V变换，以线性调频信号为例：
左图最下面为合成信号，右图为四种变换对合成信号进行的时频分析。



这里只分析利用短时傅里叶变换（Short time fourier transform, STFT）的情形。

又因为实数的傅里叶变换共轭对称，有时也仅仅分析频域的一半信息即可。

为什么要进行STFT呢，原因按我的理解可能有两点：

- 1. 传统FFT只能看到信号频率的特性，时域信号只能观察时域特性，都是一维的情况，如果二维联合观察？这个时候STFT就可以实现；
- 2. 语音是非平稳信号，比如求相关矩阵，理论上是 $E\{ \cdot \}$ 求取均值的形式，通常无法得出概率密度，往往有数据近似：

$$R_{xx} = \frac{1}{K} \sum_{i=0}^{K-1} x_n x_n^H$$

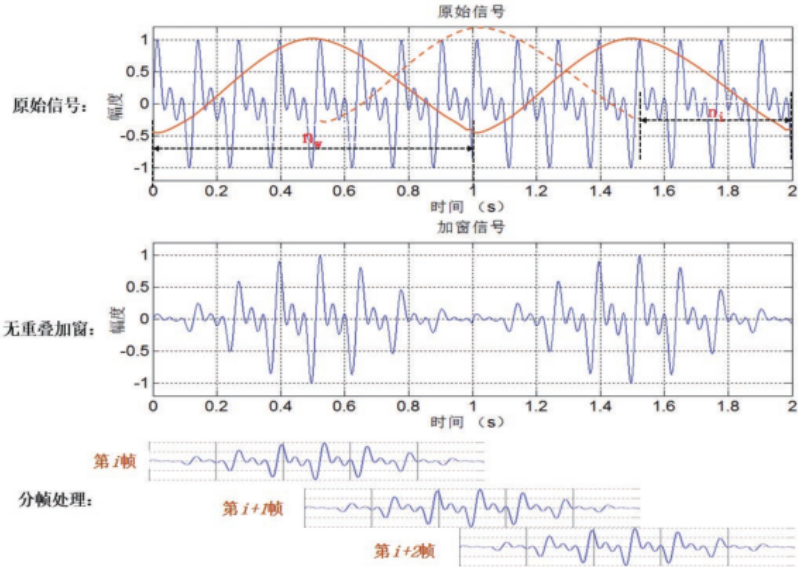
这个式子能够近似相关矩阵，有两个前提条件：a)信号平稳，这样才能保证统计特性一致；b)遍历性，这个时候才能保证统计没有以偏概全。

但语音信号是非平稳信号，直接求取相关矩阵理论上没有意义，其他统计信息也有类似的特性。但语音变化缓慢，可以认为是短时平稳，即在短的时间内（如20~30ms）是平稳的，这个时候平稳+遍历性的假设，就可以让我们借助观测数据估计统计信息。这个短时平稳的划分就是信号分帧。进一步：分帧信号分别FFT，就是STFT。

信号分帧的代码：

```
1 Nframe = floor( (length(x) - wlen) / nstep) + 1;
2 for k = 1:Nframe
3     idx = (1:wlen) + (k-1) * nstep;
4     x_frame = x(idx);
5 end
```

加窗截断分帧的示意图：



为什么要有帧移量？即帧与帧之间有部分重叠？从上面中间图可以看出，加窗阶段后相邻两帧在端点处变化较大，对于变化较大的情况一般思路就是平滑，比如进行插值处理，其实帧移的操作就是插值呀。

五、语谱图（基于FFT）

有时FFT也换成DCT实现，FFT延展与DCT是等价的，就不一一列出了。只分析FFT情况。

基于FFT语谱图的定义：

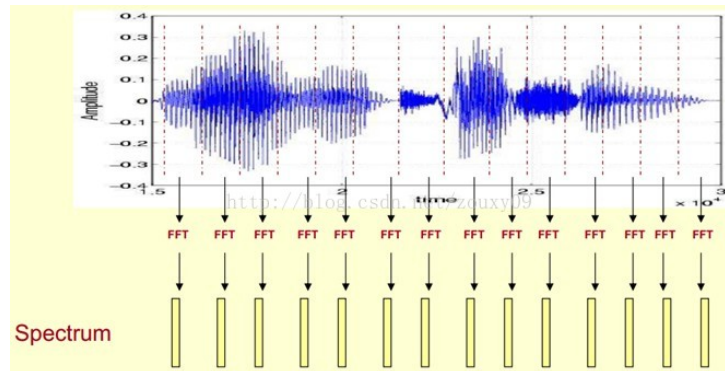
就是分帧，对每一帧信号FFT

$$X_n(e^{j\omega}) = \sum_{m=-\infty}^{\infty} x(m) \cdot w(n-m) \cdot e^{-j\omega m}$$

然后求绝对值/平方。

理论分析：

给出示意图，语音分帧→每一帧分别FFT→求取FFT之后的幅度/能量，这些数值都是正值，类似图像的像素点，显示出来就是语谱图。

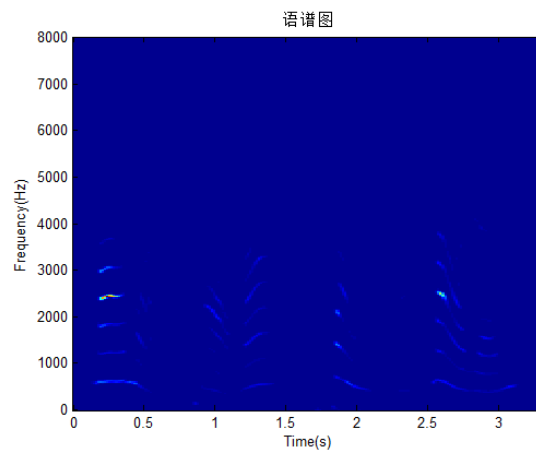


语谱图code:

```
1 function d=FrequencyCal(x,nw,ni)
2 n=nw; % 帧长
3 h=ni; % 帧移量
4 s0=length(x);
5 win=hamming(n)'; % 加窗, hamming为例
6 c=1;
7 ncols=1+fix((s0-n)/h); % 分帧, 并计算帧数
8 d=zeros((1+n/2),ncols);
9 for b=0:h:(s0-n)
10     u=win.*x((b+1):(b+n));
11     t=fft(u);
12     d(:,c)=t(1:(1+n/2))';
13     c=c+1;
14 end
```

读取语音调用语谱图code:

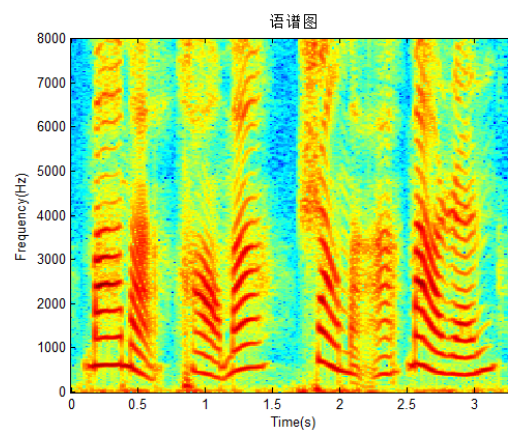
```
1 [signal,fsc] = wavread('tone4.wav');
2 nw=512;ni=nw/4;
3 d=FrequencyCal(signal',nw,ni);
4 tt=[0:ni:(length(signal)-nw)]/fsc;
5 ff=[0:(nw/2)]*fsc/nw*2;
6 % imagesc(tt,ff,20*log10(abs(d)));
7 imagesc(tt,ff,abs(d).^2);
8 xlabel('Time(s)');
9 ylabel('Frequency(Hz)');
10 title('语谱图');
11 axis xy
```



常用对数谱,修改上面的一句code即可:

```
1 imagesc(tt,ff,20*log10(abs(d)));
2 % imagesc(tt,ff,abs(d).^2);
```

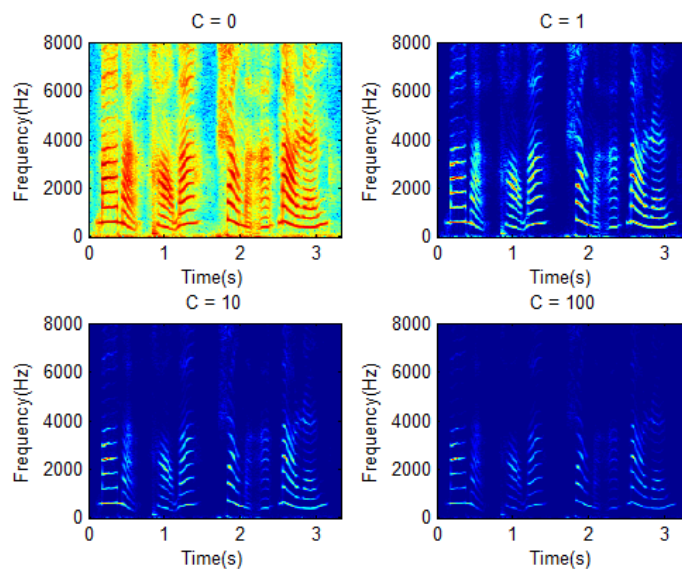
效果图:



有时对数中添加常数项;

```
1 imagesc(tt,ff,20*log10(C+abs(d)));
```

效果图:



六、短时功率谱密度

先来看看功率谱定义:

$$S_X(\omega) = \lim_{T \rightarrow \infty} \frac{1}{2T} E[|X_T(\omega)|^2]$$

可见对于有限的信号,功率谱之所以可以估计,是基于两点假设: 1) 信号平稳; 2) 随机信号具有遍历性

A-功率谱密度

1) 周期图法

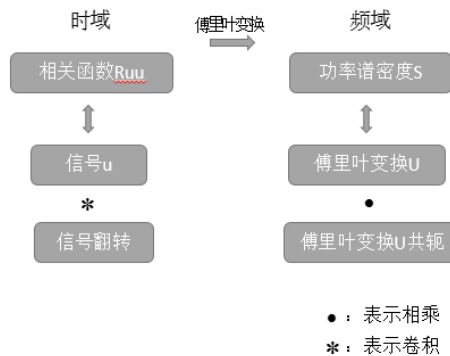
已知N个采样点的信号 $u_N(n)$ ，对其进行傅里叶变换：

$$U_N(\omega) = \sum_{n=0}^{N-1} u_N(n) e^{-j\omega n}$$

进一步得到功率谱密度：

$$\hat{S}(\omega) = \frac{1}{N} |U_N(\omega)|^2$$

按照上文分析相关函数的思路，给出一个分析：

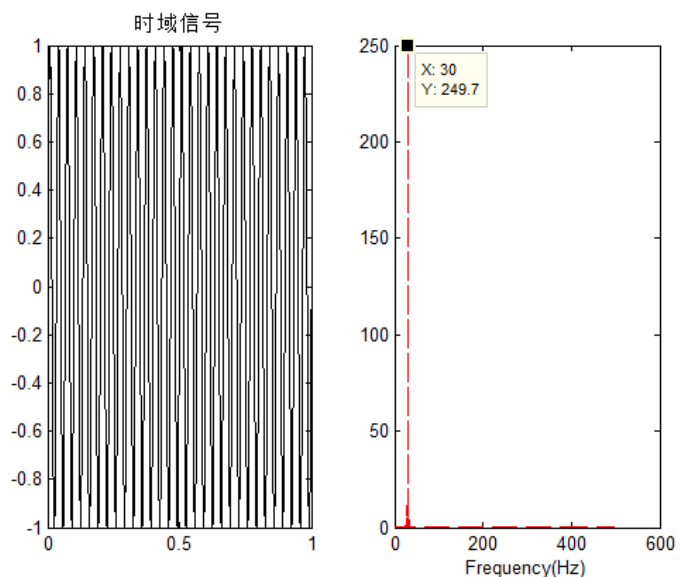


就是信号 u 的相关函数就是 u 卷积上 u 的翻转，而相关函数与功率谱密度是互为傅里叶变换。 u 对应傅里叶变换 U ， u 的翻转对应 U 的共轭，时域的卷积对应频域的相乘，就得到了功率谱估计的表达式，同样代码实现依然可以借助上文分析相关函数的特性加以分析。

对应code，其中 `my_direct_convolution` 仍然调用上面的函数：

```
1 fs = 1000; %采样率
2 f0 = 30; %信号频率
3 t = 0:1/fs:1;
4 x = sin(2*pi*f0*t);
5 lenx = length(x)
6 subplot 121
7 plot(t,x,'k');
8 title('时域信号')
9 subplot 122
10 Prr = abs(fft(my_direct_convolution(x',x(end:-1:1))))/lenx;
11 plot((0:fs)/2,Prr(1:length(Pxx)),'r--');
12 xlabel('Frequency (Hz)')
```

效果图中可以看出信号30Hz可以明显从功率谱图观察：



可以看出：功率谱密度与相关函数对应，而相关函数是统计信息，按前文提到的，它是建立在信号平稳的假设之上。如果信号不够平稳呢？周期图法的思路显然是不适用的，Welch就是对这一问题的改进。

2) 平均周期图法 (Welch)

与周期图谱求功率谱密度不同，Welch不再从整个时间段考虑，而是做了三点改进：

- 截断，将这个信号分成多个片段
- 加窗：因为截断，截断就要泄露，通常都选择加窗处理
- 重叠：截断之后，为了防止相邻两段差异过大，通常插值平滑处理，也就是取重叠

给出Welch定义，每一段的功率：

$$\hat{P}_{PER}^i(\omega) = \frac{1}{MU} \left| \sum_{n=0}^{M-1} x_N^i(n) d_2(n) e^{-j\omega n} \right|^2$$

其中 $U = \frac{1}{M} \sum_{n=0}^{M-1} d_2^2(n)$ ， $d_2(n)$ 是窗函数，M是每一段的长度，假设总长度为L，则Welch平均功率谱密度为：

$$\tilde{P}_{PER}(\omega) = \frac{1}{L} \sum_{i=1}^L \hat{P}_{PER}^i(\omega) = \frac{1}{MUL} \sum_{i=1}^L \left| \sum_{n=0}^{M-1} x_N^i(n) d_2(n) e^{-j\omega n} \right|^2$$

简单来理解就是：分段的每一段用周期图法得到功率谱密度，然后加权平均，不再细说了。

B-短时功率谱密度

按前面分析，周期图法针对的是平稳信号，而Welch虽然考虑了非平稳的特性，但分段的数量通常较小，每一段的长度较大，对音频信号而言，这也是不够的。音频信号可以看作短时间的近似平稳（如一帧信号），对每一帧利用周期图法分析，这个就是短时功率谱密度的思路。通常为了防止一帧的信号不够平稳，每一帧也可以进一步利用Welch方法处理。

对应code，分帧、Welch都是上面的思路，直接调用了：

```
1 function [Pxx] = pwelch_2(x, nwind, noverlap, w_nwind, w_noverlap, nfft)
2 % 计算短时功率谱密度函数
3 % x是信号，nwind是每帧长度，nooverlap是每帧重叠的样点数
4 % w_nwind是每帧的窗函数，或相应的段长，
5 % w_noverlap是每帧之间的重叠的样点数，nfft是FFT的长度
6
7 x=x(:);
8 inc=nwind-nooverlap; % 计算帧移
9 X=enframe(x,nwind,inc)'; % 分帧
10 frameNum=size(X,2); % 计算帧数
11 %用pwelch函数对每帧计算功率谱密度函数
12 for k=1 : frameNum
13     Pxx(:,k)=pwelch(X(:,k),w_nwind,w_noverlap,nfft);
14 end
```

七、谱熵

谱熵的定义，首先对每一帧信号的频谱绝对值归一化：

$$p_i = \frac{Y_m(f_i)}{\sum_{k=0}^{N-1} Y_m(f_k)} \quad i = 1, \dots, N$$

这样就得到了概率密度，进而求取熵：

$$H_m = - \sum_{i=0}^{N-1} p(i) \log(p(i))$$

理论分析：熵体现的是不确定性，例如抛骰子一无所知，每一面的概率都是1/6，信息量最大，也就是熵最大。如果知道商家做了手脚，抛出3的概率大，这个时候我们已经有一定的信息量，抛骰子本身的信息量就少了，熵也就变小。对于信号，如果是白噪声，频谱类似均匀分布，熵就大一些；如果是语音信号，分布不均匀，熵就小一些，利用这个性质也可以得到一个粗糙的VAD（有话帧检测）。谱熵有许多的改进思路，滤波取特定频段、设定概率密度上限、子带平滑谱熵，自带平滑通常利用拉格朗日平滑因子，这是因为防止某一段子带没有信号，这个时候的概率密度就没有意义了，这个思路在利用统计信息估计概率密度时经常用到，比如朴素贝叶斯就用到这个思路。

谱熵code：

```
1 for i=1:fn
2     Sp = abs(fft(y(:,i))); % FFT变换取幅值
3     Sp = Sp(1:wlen/2+1); % 只取正频率部分
4     Ep=Sp.*Sp; % 求出能量
5     prob = Ep/(sum(Ep)); % 计算每条谱线的概率密度
```



```

6 | H(i) = -sum(prob.*log(prob+eps)); % 计算谱熵
7 | end

```

八、基频

基频：也就是基频周期。人在发音时，声带振动产生浊音(voiced)，没有声带振动产生清音(Unvoiced)。浊音的发音过程是：来自肺部的气流冲击声门，造成声门的一张一合，形成一系列准周期的气流脉冲，经过声道（含口腔、鼻腔）的谐振及唇齿的辐射形成最终的语音信号。故浊音波形呈现一定的准周期性。所谓基音周期，就是对这种准周期而言的，它反映了声门相邻两次开闭之间的时间间隔或开闭的频率。常用的发声模型：

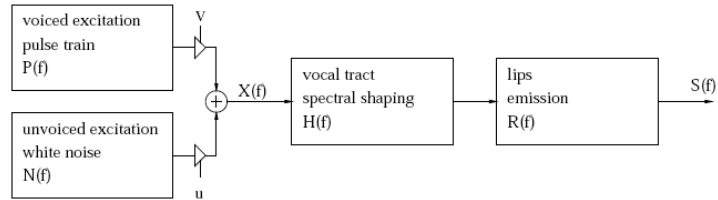


Figure 1.1: A simple model of speech production

基音提取常用的方法有：倒谱法、短时自相关法、短时平均幅度差法、LPC法，这里借用上面的短时自相关法，说一说基频提取思路。

自相关函数：

$$r_n(k) = \sum_{m=0}^{N-1-k} [x(n+m)w'(m)][x(n+m+k)w'(k+m)]$$

通常进行归一化处理，因为 $r(0)$ 最大，

$$\hat{r}_n(k) = \frac{r_n(k)}{r_n(0)}$$

得到归一化相关函数时候，归一化的相关函数第一个峰值为 $k = 0$ ，第二个峰值理论上应该对应基频的位置，因为自相关函数对称，通常取一半分析即可。

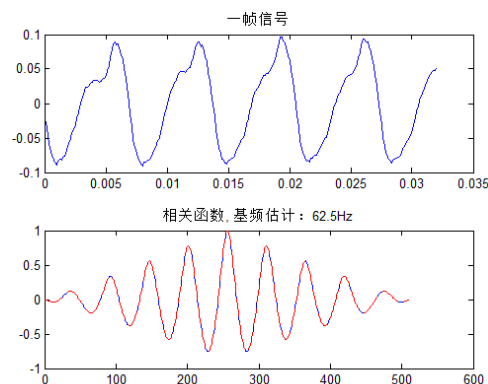
取一帧信号为例，整个说话段要配合有话音检测（VAD技术，这里不提了）自相关法估计基频code：

```

1 | [x, fs] = wavread('1.wav');
2 | nw = 256;
3 | signal = x(16001:(16000+nw),1); %取一帧信号
4 | %相关函数计算
5 | r = my_direct_convolution(signal,signal(end:-1:1)); %利用卷积计算相关函数
6 | r = r/(signal'*signal); %相关函数归一化
7 | rhalf = r(nw:end); % 取延迟量为正值的部分
8 | %提取基音，假设介于50~600Hz之间
9 | lmin = round((50/fs)*nw);
10 | lmax = round((600/fs)*nw);
11 | [tmax,tloc] = max(rhalf(lmin:lmax)); % 在Pmin~Pmax范围内寻找最大值
12 | pos = lmin+tloc-1; % 给出对应最大值的延迟量
13 | pitch = pos/nw*fs %估计得基频

```

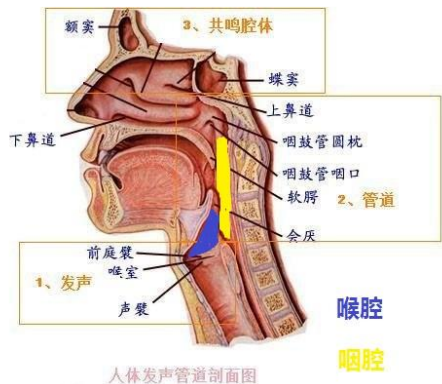
卷积还是调用上面的函数。波形的短时周期还是比较明显的，给出一帧信号：



九、共振峰

首先给出共振峰定义：当声门处准周期脉冲激励进入声道时会引起共振特性，产生一组共振频率，这一组共振频率称为共振峰频率或简称共振峰。

共振峰参数包括共振峰频率和频带的宽度，它是区别不同韵母的重要参数，由于共振峰包含在语音的频谱包络中，因此共振峰参数的提取关键是估计自然语音的频谱包络，并认为谱包括的极大值就是共振峰，通常认为共振峰数量不超过4个。发声模型：



对这个模型抽象，通常有声管模型、声道模型两个思路，以声道模型为例：认为信号经过与声道的卷积，得到最终发出的声音。声道就是系统H。

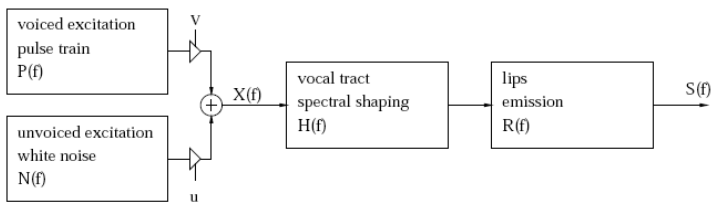


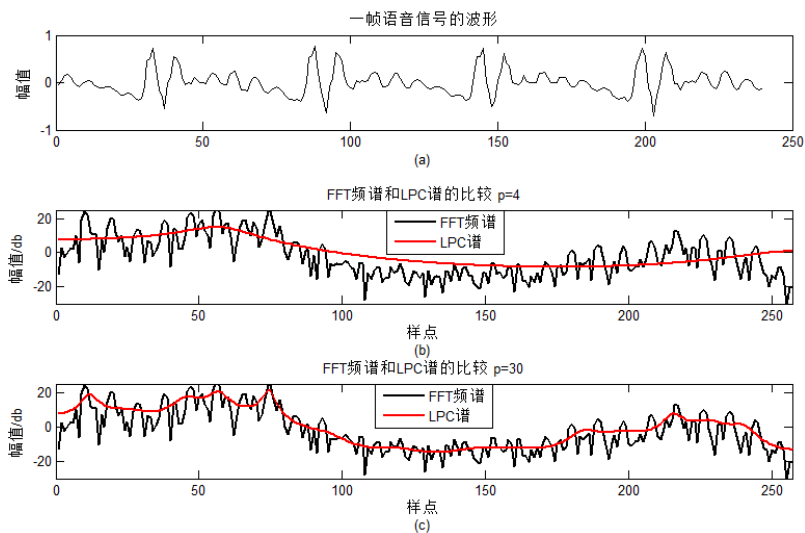
Figure 1.1: A simple model of speech production

共振峰的求解思路非常多，这里给出一个基于LPC内插的例子。

如果表达这个系统响应H呢？一个基本的思路就是利用全极点求取：

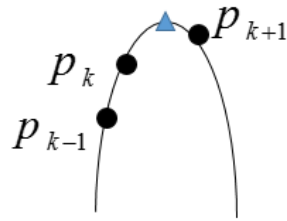
$$V(z) = \frac{G}{1 - \sum_{k=1}^N a_k z^{-k}}$$

从按照时域里信号与声道卷积的思路，频域就是信号与声道相乘，所以声道就是包络：



利用线性预测系数（LPC）并选择适当阶数，可以得到声道模型H。LPC之前有分析过，可以点击[这里](#)。

其实利用LPC得出的包络，找到四个峰值，就已经完成了共振峰的估计。为了更精确地利用LPC求取共振峰有，两种常用思路：抛物线内插法、求根法，这里以内插法为例。其实就是一个插值的思路，如图



为了估计峰值，取 $p(k-1)$ 、 $p(k)$ 、 $p(k+1)$ 哪一个点都是不合适的，插值构造抛物线，找出峰值就更精确了。

LPC内插法code:

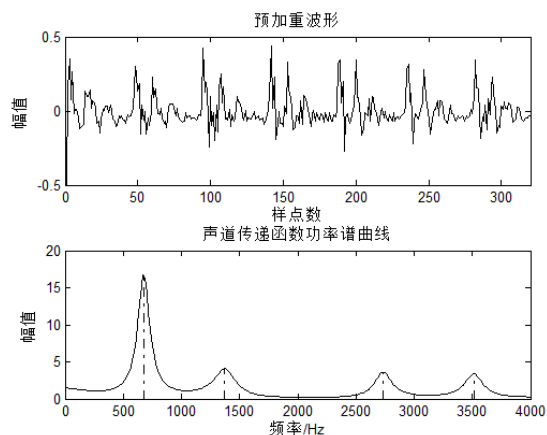
```

1  [x,fs]=wavread(file);           % 读入一帧语音信号
2  u=filter([1 -.99],1,x);         % 预加重
3  wlen=length(u);                 % 帧长
4  p=12;                           % LPC阶数
5  a=lpc(u,p);                     % 求出LPC系数
6  U=lpcar2pf(a,255);              % 由LPC系数求出频谱曲线
7  freq=(0:256)*fs/512;            % 频率刻度
8  df=fs/512;                      % 频率分辨率
9  U_log=10*log10(U);               % 功率谱分贝值
10 [Loc,Val]=findpeaks(U);          % 在U中寻找峰值
11 ll=length(Loc);                  % 有几个峰值
12 %抛物线内插修正
13 for k=1 : ll
14     m=Loc(k);                     % 设置m-1, m和m+1
15     m1=m-1; m2=m+1;
16     p=Val(k);                     % 设置P(m-1), P(m)和P(m+1)
17     p1=U(m1); p2=U(m2);
18     aa=(p1+p2)/2-p;
19     bb=(p2-p1)/2;
20     cc=p;
21     dm=-bb/2/aa;
22     pp=-bb*bb/4/aa+cc;
23     m_new=m+dm;
24     bf=-sqrt(bb*bb-4*aa*(cc-pp/2))/aa;
25     F(k)=(m_new-1)*df;
26     Bw(k)=bf*df;
27 end

```

为了更好地估计声道，这里用了预加重，因为类似电磁波等信号，波信号传播过程中高频分量衰减更大，所以需要高频进行一定程度的提升，这个操作叫做：预加重。

对应的效果图：



求解的共振峰频率 (Hz) :676.15 1372.53 2734.15 3513.69, 基音周期与共振峰不是一回事啊！有时为了表征声道特性，也可以直接利用LPC系数作为特征，而不必求取共振峰。

梅尔倒谱系数(MFCC)打算单拎出来写了，涉及的概念有点多，其他特征用到再补充吧，[上一篇文章](#)也提到了很多音频特征的概念。

参考：

- 宋知用《MATLAB在语音信号分析与合成中的应用》

分类: [20-音频信号](#)

标签: [音频信号](#), [LPC](#), [过零率](#), [基音周期](#), [相关函数](#)



桂。
关注 - 15
粉丝 - 260

[+加关注](#)

« 上一篇: [2017随记——五月](#)

» 下一篇: [统计学习方法：决策树（1）](#)

posted @ 2017-05-06 08:17 桂。 阅读(8677) 评论(3) 编辑 收藏

评论列表

#1楼 2017-05-06 22:09 自由布鲁斯

学习一下

支持(0) 反对(0)

#2楼 2017-05-06 22:13 MJTuc.Di`

不明觉厉!

支持(0) 反对(0)

#3楼 2017-05-07 00:04 低头赏月

一脸懵逼地进来，又一脸懵逼地出去。

支持(0) 反对(0)

[刷新评论](#) [刷新页面](#) [返回顶部](#)

注册用户登录后才能发表评论，请 [登录](#) 或 [注册](#)，[访问网站首页](#)。

【推荐】超50万VC++源码：大型组态工控、电力仿真CAD与GIS源码库！

【免费】要想入门学习Linux系统技术，你应该先选择一本适合自己的书籍

【前端】SpreadJS表格控件，可嵌入应用开发的在线Excel

【直播】如何快速接入微信支付功能



最新IT新闻：

- 少儿编程市场的冰与火
- Visual Studio Code 1.27.2发布，Bug修复版
- OpenStack的八年之痒
- 库克谈最贵iPhone定价：够创新所以很合理
- AMD或于本月底前宣布对32bit显卡驱动停更
- » 更多新闻...



华为全联接大会 | 上海 | 2018.10.10-12
[大会门票+云服务器] 专属套餐0.35折起



最新知识库文章：

- 为什么说 Java 程序员必须掌握 Spring Boot ?
- 在学习中，有一个比掌握知识更重要的能力
- 如何招到一个靠谱的程序员
- 一个故事看懂“区块链”
- 被踢出去的用户

» 更多知识库文章...