# 特征比较

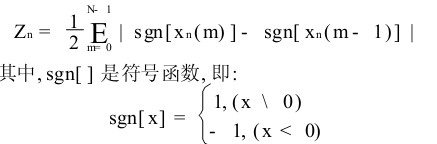
我们除了mel特征，fft特征外还提取了如下几个特征：过零率、短时能量、短时平均幅度差、谱熵、自相关函数、语谱，下面对这些特征进行简要介绍，代码文件feat.c

## 过零率：

### 定义：

归一化后一帧内穿越0的次数

### 公式：



### C代码：

//首先归一化

double max = tmp[0],min = tmp[0];

for(i=0;i<FRAME\_SIZE;i++){

max = tmp[i]>max? tmp[i]:max;

min = tmp[i]<min? tmp[i]:min;

}

//统计过零个数

double zcr = 0;

double norma = 0.5\*(max+min);

for (i = 0; i<FRAME\_SIZE-1; i++){

zcr += (tmp[i]-norma)\*(tmp[i+1]-norma)<0? 1:0;

}

fprintf(f2,"过零率%f\n",zcr/FRAME\_SIZE);

### 分析：

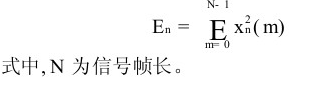
是时域的音频特征，通过统计信号通过零点次数体现信号频率特性。提取特征算法复杂度为

## 短时能量：

### 定义：

一帧内信号的平方和。一般采用对数能量，即对结果取对数\*10.

### 公式：



### C代码：

double sum = 0;

for(i = 0; i < FRAME\_SIZE ; i++){

sum += tmp[i]\*tmp[i];

}

fprintf(f2,"对数能量%f\n",log10(sum)\*10);

### 分析：

是时域的音频特征，体现信号在不同帧的强弱程度。提取特征算法复杂度为

## 短时平均幅度差：

### 定义：

所有点与其他所有点的幅度差的绝对值和的均值。

### 公式：

https://images2015.cnblogs.com/blog/1085343/201705/1085343-20170505224104554-1592531899.png

### C代码：

double amdvec = 0;

for(i = 0; i < FRAME\_SIZE ; i++){

for(int j = i; j<FRAME\_SIZE; j++){

amdvec += std::abs(tmp[i]-tmp[j]);

}

}

fprintf(f2,"短时平均幅度差 %f\n",amdvec);

### 分析：

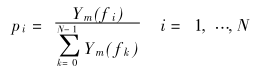
是时域的音频特征。平稳噪声情况下利用短时平均幅度差可以更好地观察周期特性。 提取特征算法复杂度为 不建议使用

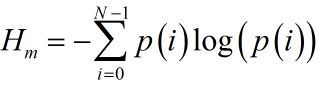
## 谱熵：

### 定义：

FFT变换后得到的频谱的熵。

### 公式：





### C代码

double ep[FRAME\_SIZE],probi,sumep = 0,H = 0;//H是熵 每帧一个

p=fftw\_plan\_dft\_1d(FRAME\_SIZE,in,out,FFTW\_FORWARD,FFTW\_MEASURE);

for(i=0;i<FRAME\_SIZE;i++){

in[i][0]=tmp[i];

in[i][1]=0.0;

}

fftw\_execute(p);

for (i=0;i<FRAME\_SIZE;i++){

ep[i] = out[i][0]\*out[i][0]+out[i][1]\*out[i][1];

sumep += ep[i];

}

for (i =0;i<FRAME\_SIZE ;i++){

probi = ep[i]/sumep;

H += probi\*log(probi)\*(-1);

}

fprintf(f2,"谱熵%f\n",H);

### 分析：

是频域的音频特征，体现信号在频域的分布情况。是对FFT结果的统计。分布越随机，谱熵越大。提取特征算法复杂度主要是FFT,因此为

## 自相关函数：

### 定义：信号在时域和自身的卷积

### 公式：

https://images2015.cnblogs.com/blog/1085343/201705/1085343-20170505220719867-1026965029.png

我们先计算其在频域的乘积后，进行IFFT。

### C代码：

for(i=0;i<FRAME\_SIZE; ++i){

out[i][0] = ep[i]/FRAME\_SIZE;

out[i][1] = 0.0;

}

p = fftw\_plan\_dft\_1d(FRAME\_SIZE,out,in,FFTW\_BACKWARD,FFTW\_MEASURE);

fftw\_execute(p);

int n\_size = FRAME\_SIZE/8;

double sc[8];

fprintf(f2,"自相关");

for(i=0;i<8;++i){

int j = i\*n\_size;

sc[i]=(in[j][0]);

fprintf(f2,"%f,",sc[i]);

}

fprintf(f2,"\n");

### 分析：

由于自相关函数得到的是一个序列，因此等距抽样为8个点作为特征，复杂度受傅里叶变换的影响

## 语谱：

### 定义：

语音信号的傅里叶分析的显示图形称为语谱图

### 公式：

https://images2015.cnblogs.com/blog/1085343/201705/1085343-20170505231535836-635606981.png，求出的FFT，绝对值除以平方。

### C代码：

double sg[8];

fprintf(f2,"语谱");

for(i=0;i<8;++i){

int j = i\*n\_size;

sg[i]=(100000.0/ep[j]);//等比例扩大十万倍，方便区分

fprintf(f2,"%f,",sg[i]);

}

fprintf(f2,"\n");

### 分析：

频域内的特征，同样进行了抽样，其实和FFT本身特征相似，复杂度O(1),前提是利用之前FFT的结果。