# 预处理函数结构和示例

# 对齐模块

对齐模块的程序结构图如图6-1所示。



图6-1 对齐模块类关系图

对于不同的对齐方法都通过Alignment\_Mthod父类进行统一，每个子类实现时都必须继承父类的接口函数，接口函数如表6-1所示。

表6-1 Alignment\_Method父类接口函数说明

|  |  |
| --- | --- |
| 接口函数名 | 功能描述 |
| vector<double> Align(vector<double>) | 输入功耗曲线，返回对齐后功耗曲线； |

通过工厂类Alignment\_Factory对指定算法进行实例，每个子类中实现父类接口。

## 6.1 静态对齐

对于只在时域上未对齐的曲线，静态对齐分别计算出待对齐的曲线和模板曲线之间的错位量，再把待对齐曲线进行相应的平移。计算错位量时，设置一个滑动窗口，依次计算从滑动窗口中每个点开始的曲线和模板曲线的相关系数，再选择相关系数最大的点，它在滑动窗口中的索引值即为错位量。

算法说明如算法6-1所示。

|  |
| --- |
| Algorithm 6-1: CORRELATION |
| Input: X //参照曲线  Y //待对齐曲线  window //滑动窗口大小，一般通过观察一条曲线相邻峰值的范围和两条曲线之间的偏移量来估算  Output: Y’ //将Y向X对齐之后的曲线 |
| for i=0:window-1  cor[i]=corr(X,leftShift(Y,i)); // 分别计算X与Y循环左移i个点之后的曲线的相关系数，存于cor矩阵  cortmp=0;//滑动窗口中的最大相关系数  shift=0; //最大相关系数对应的偏移量  for i=0:window-1; //寻找滑动窗口中最大的相关系数并记录对应偏移量  if(cor[i]>cortmp){  cortmp=cor[i];  shift=i; }  Y’=leftShift(Y,shift) ; //将Y曲线循环左移shift个点 |

除继承父类函数接口，静态对齐类主要成员类及成员函数说明如表6-2所示。

表6-2 静态对齐类成员说明

|  |  |
| --- | --- |
| 成员名 | 功能 |
| vector<double> base\_trace; | 基准曲线 |
| int start; | 扫描区间的起点 |
| int end; | 扫描区间的终点 |
| int RangeOffset; | 扫描区间的范围。 |
| double Cal\_Correlation(vector<double > p0, vector<double > p1, int pointnum); | 计算两条曲线的相关性，pointnum为曲线的点数 |

## 6.2 DTW

DTW（Dynamic Time Warping）[JMB11]为动态时间规整算法，它是语音识别中用于孤立词识别的算法。DTW是把时间规整和距离测量计算结合起来的一种非线性规整技术，它不断计算两条曲线上时间点的距离以寻找最优的匹配路径，所以得到的是两条曲线匹配时累积距离最小所对应的规整函数，这就保证了两条曲线之间存在最大相似性。DTW算法的实质就是运用动态规划的的思想，利用局部最佳化的处理来自动寻找一条路径，沿着这条路径两条曲线之间的累积失真最小，从而避免时长不同而可能引入的误差。

DTW算法分三步进行，一是计算两条曲线个点之间的距离，即求出点匹配距离矩阵，二是在点匹配距离矩阵中找出一条最佳路径，三是按最佳路径将待对齐曲线按照最佳路径想模板曲线进行对齐。

DTW算法描述如下。

|  |
| --- |
| Algorithm 6-2: Dynamic Time Warping |
| Input: X //参照曲线  Y //待对齐曲线  Output: Y’ //将Y向X对齐之后的曲线 |
| 1. d(i,j)=abs(Y[j]-X[i]) //距离函数，分别计算Y中每个点到X中每个点的绝对距离，存于d中  2. g(i,j)=min[g(i,j-1)+d(i,j),g(i-1,j)+d(i,j),g(i-1,j-1)+2d(i,j)] //代价函数，分别计算Y中每个点到X中每个点的代价，存于g中  3. path=warp(g) // 根据g矩阵寻找代价最小的路径存于path中  4. Y’=Y(path) //根据代价最小路径将Y曲线进行对齐 |

除继承父类函数接口外，DTW类主要成员类及成员函数说明如表6-3所示。

表6-3 DTW类成员说明

|  |  |
| --- | --- |
| 成员名 | 功能 |
| vector<double> base\_trace; | 基准曲线 |
| vector<vector<double>> distance | 保存距离 |
| vector<vector<int>> dtwpath | 保存路径 |
| vector<vector<int>> F; | 保存路径 |
| void DTWDistanceFun(vector<double> A,int I, vector<double> B,int J,int r); | 求两个数组之间的匹配距离，A,B分别为第一第二个数组，I，J为其数组长度，r为匹配窗口的大小，r的大小一般取为数组长度的1/10到1/30；返回两个数组之间的匹配距离,如果返回－1，表明数组长度太大了 |

## 6.3 POC

POC（Phase Only Correlation）[NST08]为图像匹配中广泛使用的一种算法，它具有高的噪声容忍度、失真小等优点。它是一种非线性、基于傅里叶功率谱的频域相关算法，由于采用该方法只取功率谱中的相位信息，因而减少了对曲线的依赖性，而且所获得的相关性峰值较大，因此具有很高的匹配精度。

利用POC对曲线进行对齐时，先对曲线作傅里叶变换将曲线转换到频域。由傅里叶移位定理可知，时域上的移位等价于频域上的移位。因此，POC计算待对齐曲线和模板曲线之间的相位差，再在频域上进行对齐，最后作逆傅里叶变换转回时域。

算法说明如算法6-3所示。

|  |
| --- |
| Algorithm 6-3: Phase Only Correlation |
| Input: ga //参照曲线，由于需要做傅里叶变换，曲线点数应为2的幂  gb //待对齐曲线，由于需要做傅里叶变换，曲线点数应为2的幂  Output: gb’ //gb向ga对齐之后的曲线 |
| 1. Ga=F(ga); Gb=F(gb); //分别对两条曲线进行傅里叶变换(FFT)  2.  //计算交叉能量频谱([cross-power spectrum](https://en.wikipedia.org/w/index.php?title=Cross-power_spectrum&action=edit&redlink=1)),为的共轭  3. r=F-1(R) //对R做逆离散傅里叶变换(IFFT)  4.  //对r进行函数拟合,算出α和δ,N为曲线点数  5. //根据傅里叶移位定理,时域的移位对应于频域的相位移位，k=0...N-1，i为虚数单位  6. //将频域对齐的曲线转换到时域 |

除继承父类函数接口外，POC类主要成员类及成员函数说明如表6-4所示。

表6-4 POC类成员说明

|  |  |
| --- | --- |
| 成员名 | 功能 |
| vector<double > base\_trace; | 基准曲线 |
| int Points | 处理点数 |
| vector<complex>fft(vector<complex> ) | 快速傅里叶变换函数 |
| vector<complex>ifft(vector<complex> ) | 逆傅里叶变换函数 |
| complex Diexing\_operator(complex, complex) | 蝶形变换函数 |

## 6.4 FFT

FFT（Fast Fourier Transform）为快速傅里叶变换，是离散傅里叶变换（DFT，Discrete Fourier Transform）的快速算法。大多数离散时间信号来源于连续时间信号的抽样，即利用周期性的脉冲序列从连续时间信号中抽取一系列等间隔的离散值得到。抽样间隔大时，得到的抽样信号不能反映原连续时间信号的细节；抽样间隔小时，需要处理太多的抽样点。

在抽样过程中，抽样频率的大小与连续信号变化快慢具有一定的关系。Nyquist抽样定理给出了最小的抽样频率，要想对离散信号不失真地还原模拟信号，抽样频率必须大于等于原信号谱的最高频率。

示波器采集的数据为有限长的离散信号，故需要采用离散傅里叶变换来进行刻画。根据傅里叶移位定理，信号在时域的移位等价于频域中相位的移位。

在功耗、电磁分析中，示波器采集的数据可看作能量有限信号，时域信号的能量等价于频域信号的能量，即信号经傅里叶变换，其总能量保持不变，符合能量守恒定律。在对加入了时延防护的实现进行分析时，通常需要先利用FFT将时域信号转化为频域信号，在消除时延再进行密钥分析。

FFT算法流程图如6-2所示，除继承父类函数接口外，FFT类主要成员函数说明如表6-5所示。

表6-5 FFT类成员函数说明

|  |  |
| --- | --- |
| 函数名 | 功能 |
| vector<complex> YW\_to\_ComplexYW(vector<double>,int) | double类型数据转换为Complex类型； |
| vector<complex> function\_fft(vector<complex>, int) | 对数据执行FFT，供内部调用； |
| complex Diexing\_operator(complex a, complex b); | 蝶形变换函数 |
| vector<double> ABS(vector<complex>,int ); | 计算复数的模； |
| vector<double> Select\_Samples(vector<double> data, vector<int> index, int select\_Num); | 选择样本点； |
| vector<int> Select\_Samples\_Index(vector<double>&data, int select\_Num); | 选择样本点下标； |



图6-2 FFT算法流程图

# 3 对齐操作示例

对齐方法包括：StaticAlignment（静态对齐）、DTW（动态时间规整）、POC（相位相关）、FFT（快速傅立叶变换）。操作示例过程如下。

（1）路径设置与2定位操作示例相同。

（2）点击菜单栏“分析”按钮，点击下拉菜单中“对齐”。设置对齐方法、扫描偏移量（当选静态对齐时设置）、起始点、点数、总点数（当选静态对齐时设置）、曲线数量。



图3-1对齐方法示例面板选项设置

以DTW为例，选择DTW，设置起始点为0，点数为1024，曲线量设置为10。**值得说明的是，FFT和POC对齐方法要求点数必须为2的整数次幂，且FFT后曲线点数为原始点数的一半。**

（3）点击确定按钮，开始处理，处理完成后在存储曲线的目录下生成对齐后的曲线，命名规则为对齐方法+系统时间。同时在界面显示对齐后的两条曲线，如图3-2所示。

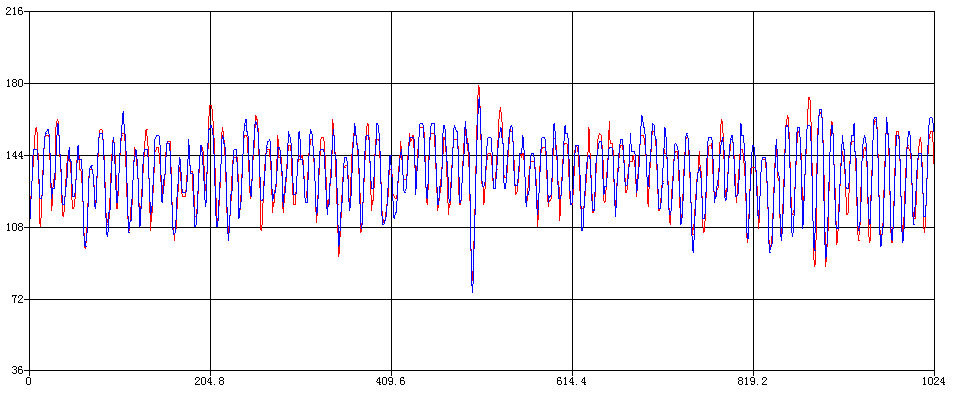


图3-2 DTW对齐后曲线