南開大學
Nankai University

网络空间安全学院

信息隐藏技术实验报告

语音信号的常用处理

姓名：孙蕗

学号：2112060

专业：信息安全

2024 年 3 月 17 日

目录

1 实验内容

学习慕课 2.2 语音信号处理基础，分别使用 FFT、DWT、DCT 三种方法进行语音信号处理

2 原始音频

对实验中要用到的语音音频文件进行读取及展示

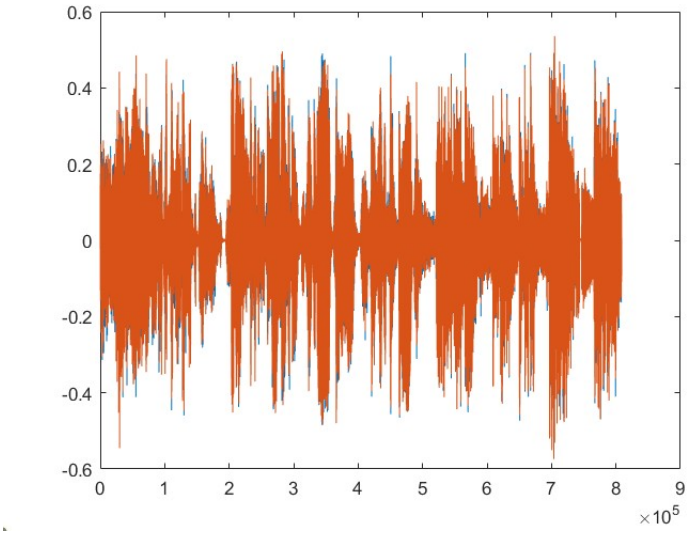


图 2.1: original.jpg

Listing 1: 原始音频文件读取

```
1
2 [x,fs] = audioread ('testaudio.wav') ;
3 plot(x) ;
```

3 FFT 快速傅里叶变换

3.1 实验原理

快速傅里叶变换 (Fast Fourier Transform)，即利用计算机计算离散傅里叶变换（DFT）的高效、快速计算方法的统称，简称 FFT。对多项式 $f(x)=\sum_{i=0}^{n}a_ix^i$ ， $g(x)=\sum_{i=0}^{n}b_ix^i$，定义其乘积 fg 为 $(fg)(x)=(\sum_{i=0}^{n}a_ix^i)(\sum_{i=0}^{n}b_ix^i)$。

显然可以以 $O(n^2)$ 的复杂度计算这个乘积的每一项的系数。但 FFT 可以以 $O(nlogn)$ 的时间复杂度来计算这个乘积。

3.1.1 多项式的表示

1. 多项式的系数表示

对于任意 $n+1$ 次多项式 $f(x)=\sum_{i=0}^{n}a_ix^i\in X_n$，只需知道每一项的系数就可以唯一确定 $f(x)$，故 $f(x)$ 可以被写作 $\{a_0,a_1,a_2,...,a_n\}$ 的形式，即其系数的有序排列，这就是多项式的系数表示。

2. 多项式的点值表示

因为 n 次多项式上 $n+1$ 个不同的点能唯一确定这个多项式, 即将 $n+1$ 个不同的数 $\{a_0, a_1, a_2, ..., a_n\}$ 带入 $f(x)$，就能得到 $n+1$ 个不同的数值 $y_i = f(x_i)$，进而存在唯一满足 $\forall 0 \le i \le n, f(x_i) = y_i$ 的 n 次多项式 f。把这 n 条式子联立起来成为一个有 n 条方程的 n 元方程组，每一项的系数都可以解出来。上述性质可以通过因式定理与代数基本定理或者范德蒙行列式的性质证明。

因此使用 $(x_0, f(x_0)), (x_1, f(x_1)), ..., (x_n, f(x_n))$ 就可以完整描述出这个多项式 $f(x)$，这就是多项式的点值表示。

当使用点值表示法表示多项式的相乘时，如果两个多项式取相同的 x，得到不同的 y 值，那么只需要 y 值对应相乘就可以了, 复杂度只有枚举 x 的 $O(n)$。所以如果能够在较低的时间复杂度内将系数表示法转化为点值表示法，再将点值表示法转回系数表示法，就能以较低的时间复杂度计算多项式的乘法。

3.1.2 DFT 离散傅里叶变换

以单位圆点为起点，单位圆的 n 等分点为终点，在单位圆上可以得到 n 个复数，设幅角为正且最小的复数为 ω_n ，称为 n 次单位根，即 $\omega_n = \cos\frac{2\pi}{n} + \mathrm{i}\sin\frac{2\pi}{n}$

由欧拉公式 $\omega_n^k = \cos\frac{2k\pi}{n} + \mathrm{i}\sin\frac{2k\pi}{n}$。特别地，$\omega_n^0 = \omega_n^n = 1$ 。

可以发现单位根具有以下性质:

$$\omega_{rn}^{rk} = \cos\frac{2rk\pi}{rn} + \mathrm{i}\sin\frac{2rk\pi}{rn} = \omega_n^k \qquad r \in \mathbb{N}^+ \tag{1}$$

$$\omega_n^{k+\frac{n}{2}} = \omega_n^k \cdot \left(\cos\left(\frac{n}{2}\cdot\frac{2\pi}{n}\right) + \mathrm{i}\sin\left(\frac{n}{2}\cdot\frac{2\pi}{n}\right)\right) \qquad k \in \mathbb{N}^+ \tag{2}$$

$$= \omega_n^k \cdot (\cos\pi + \mathrm{i}\sin\pi) \tag{3}$$

$$= -\omega_n^k \tag{4}$$

$$\overline{\omega_n^k} = \cos\frac{2k\pi}{n} - \mathrm{i}\sin\frac{2k\pi}{n} \tag{5}$$

$$= \cos\left(2\pi - \frac{2k\pi}{n}\right) + \mathrm{i}\sin\left(2\pi - \frac{2k\pi}{n}\right) \tag{6}$$

$$= \omega_n^{n-k} \tag{7}$$

DFT 其实就是把上述的 n 个复数（单位根）$\omega_n^0, \omega_n^1, ..., \omega_n^n - 1$ 代入多项式，能得到一种特殊的点值表示，这种点值表示就叫 DFT（离散傅里叶变换)。

3.1.3 FFT 快速傅里叶变换

对多项式 $f(x) = \sum_{i=0}^{n-1} a_i x^i \in X_{n-1}$ ，不失一般性的，设 $n = 2^s \quad s \in \mathbb{N}$ （由于在多项式的乘法中, 可以将一个多项式等价地看作是次数更高的高次项系数均为零的多项式，故可以将 n 看作第一个大于等于它的 2 的整数次幂)，考虑按 a_i 下标的奇偶性将 $f(x)$ 中的项分为两部分，即

$$f(x) = (a_0 + a_2x^2 + a_4x^4 + \cdots + a_{n-2}x^{n-2}) \tag{8}$$
$$+ (a_1x + a_3x^3 + a_5x^5 + \cdots + a_{n-1}x^{n-1}) \tag{9}$$

令

$$f_1(x) = a_0 + a_2x^1 + a_4x^2 + \cdots + a_{n-2}x^{\frac{n}{2}-1} \tag{10}$$
$$f_2(x) = a_1 + a_3x + a_5x^2 + \cdots + a_{n-1}x^{\frac{n}{2}-1} \tag{11}$$

则

$$f(x) = f_1(x^2) + xf_2(x^2)$$

带入 $x = \omega_n^k \quad (k < \frac{n}{2})$ 可得

$$f(\omega_n^k) = f_1(\omega_n^{2k}) + \omega_n^k f_2(\omega_n^{2k}) \tag{12}$$
$$= f_1(\omega_{\frac{n}{2}}^k) + \omega_n^k f_2(\omega_{\frac{n}{2}}^k) \tag{13}$$

带入 $x = \omega_n^{k+\frac{n}{2}} \quad (k < \frac{n}{2})$ 可得

$$f(\omega_n^{k+\frac{n}{2}}) = f_1(\omega_n^{2k+n}) + \omega_n^{k+\frac{n}{2}} f_2(\omega_n^{2k+n}) \tag{14}$$
$$= f_1(\omega_n^{2k} \cdot \omega_n^n) - \omega_n^k f_2(\omega_n^{2k} \cdot \omega_n^n) \tag{15}$$
$$= f_1(\omega_n^{2k}) - \omega_n^k f_2(\omega_n^{2k}) \tag{16}$$
$$= f_1(\omega_{\frac{n}{2}}^k) - \omega_n^k f_2(\omega_{\frac{n}{2}}^k) \tag{17}$$

观察 (12)-(13),(14)-(17) 两式的结构，只需要求出 $f_1(\omega_{\frac{n}{2}}^k), f_2(\omega_{\frac{n}{2}}^k)$ 即可 O(1) 求出 (1),(2) 两式的值，进而再经过类似的步骤，可以以 O(1) 的时间复杂度将问题继续转化为求 $f_1(\omega_{\frac{n}{4}}^k), f_2(\omega_{\frac{n}{4}}^k)$ ……最终问题被转化为求

$$f_1(\omega_1^k) = f_2(\omega_1^k) = 1$$

故以 $O(\log n)$ 的时间复杂度可以求出 $f(\omega_n^k), f(\omega_n^{k+\frac{n}{2}})$，进而可以以 $O(n \log n)$ 的时间复杂度求出所有的 $f(\omega_n^k)$ ，即以 $O(n \log n)$ 的时间复杂度将 f 的系数表示法转化为了点值表示法。

3.1.4 IDFT 快速傅里叶逆变换

跑完 FFT 后就得到了多项式乘积的点值表示，现在需要将点值表示转回系数表示，这个转换的过程被称为离散傅里叶逆变换（IDFT）。

如果用矩阵将 DFT 的过程封装，那么 DFT 就相当于求

$$\begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & (\omega_n^1)^1 & (\omega_n^1)^2 & \cdots & (\omega_n^1)^{n-1} \\ 1 & (\omega_n^2)^1 & (\omega_n^2)^2 & \cdots & (\omega_n^2)^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & (\omega_n^{n-1})^1 & (\omega_n^{n-1})^2 & \cdots & (\omega_n^{n-1})^{n-1} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{n-1} \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_{n-1} \end{bmatrix} \tag{18}$$

其中

$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{n-1} \end{bmatrix}$ 意义为多项式的系数表示法，

$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_{n-1} \end{bmatrix}$ 意义为多项式的点值表示法，y_i 即为 $f(\omega_n^i)$ 。

注意到

$$x^n - 1 = (x-1)(x^{n-1} + x^{n-2} + \cdots + 1)$$

于是在 $x = 1$ 时

$$x^{n-1} + x^{n-2} + \cdots + 1 = n$$

否则在

$$x^n - 1 = 0 \; x^{n-1} + x^{n-2} + \cdots + 1 = 0$$

其中 x 为 n 次单位根。

受 n 次单位根这一特性的启发，注意到 $x^{n-1} + x^{n-2} + \cdots + 1$ 在其形式上与矩阵乘法的关系，可

以发现

$$\begin{bmatrix} 1 & (\omega_n^{n-i})^1 & (\omega_n^{n-i})^2 & \cdots & (\omega_n^{n-i})^{n-1} \end{bmatrix} \begin{bmatrix} 1 \\ (\omega_n^j)^1 \\ (\omega_n^j)^2 \\ \vdots \\ (\omega_n^j)^{n-1} \end{bmatrix} = \sum_{k=0}^{n-1} (\omega_n^{n-i})^k (\omega_n^j)^k \tag{19}$$

$$= \sum_{k=0}^{n-1} (\omega_n^{n-i+j})^k \tag{20}$$

$$= \begin{cases} n & i = j \\ 0 & i \neq j \end{cases} \tag{21}$$

这表明

$$\mathbf{C} = \frac{1}{n} \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & (\omega_n^{n-1})^1 & (\omega_n^{n-1})^2 & \cdots & (\omega_n^{n-1})^{n-1} \\ 1 & (\omega_n^{n-2})^1 & (\omega_n^{n-2})^2 & \cdots & (\omega_n^{n-2})^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & (\omega_n^1)^1 & (\omega_n^1)^2 & \cdots & (\omega_n^1)^{n-1} \end{bmatrix} \tag{22}$$

为

$$\begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & (\omega_n^1)^1 & (\omega_n^1)^2 & \cdots & (\omega_n^1)^{n-1} \\ 1 & (\omega_n^2)^1 & (\omega_n^2)^2 & \cdots & (\omega_n^2)^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & (\omega_n^{n-1})^1 & (\omega_n^{n-1})^2 & \cdots & (\omega_n^{n-1})^{n-1} \end{bmatrix} \tag{23}$$

的逆矩阵。

由上文部分

$$\mathbf{C} = \frac{1}{n}\begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & (\omega_n^{n-1})^1 & (\omega_n^{n-1})^2 & \cdots & (\omega_n^{n-1})^{n-1} \\ 1 & (\omega_n^{n-2})^1 & (\omega_n^{n-2})^2 & \cdots & (\omega_n^{n-2})^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & (\omega_n^{1})^1 & (\omega_n^{1})^2 & \cdots & (\omega_n^{1})^{n-1} \end{bmatrix} \tag{24}$$

$$= \frac{1}{n}\begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & (\overline{\omega_n^{1}})^1 & (\overline{\omega_n^{1}})^2 & \cdots & (\overline{\omega_n^{1}})^{n-1} \\ 1 & (\overline{\omega_n^{2}})^1 & (\overline{\omega_n^{2}})^2 & \cdots & (\overline{\omega_n^{2}})^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \overline{(\omega_n^{n-1})}^1 & \overline{(\omega_n^{n-1})}^2 & \cdots & \overline{(\omega_n^{n-1})}^{n-1} \end{bmatrix} \tag{25}$$

$$\tag{26}$$

所以将一个多项式在分治的过程中乘上的单位根变为其共轭复数，分治完的每一项除以 n 即可得到原多项式的每一项系数。

3.2 Matlab 源码

Listing 2: 快速傅里叶变换

```
1
2  %快速离散傅里叶变换
3  [x,fs]=audioread('testaudio.wav');
4  fx=fft(x);%fft函数 快速离散傅里叶变换
5
6  x0=ifft(fx);%逆变换
7
8  figure('name','快速傅里叶变换FFT');
9
10 subplot(3,1,1);
11 plot(x);
12 title('Raw Audio')
13
14 subplot(3,1,2);
15 plot(abs(fftshift(fx)));
16 xlabel('Time');
17 ylabel('Amplitude');
18 title('The Wave form of signal FFT');
19
20 grid on;
21
22 subplot(3,1,3);
23 plot(x0);
```

```
24 title('Restored audio')
```

3.2.1 Matlab 中的 FFT 函数

- 语法

  Y = fft(X)

  Y = fft(X,n)

  Y = fft(X,n,dim)

- 说明

  Y = fft(X) 使用快速傅里叶变换 (FFT) 算法计算 X 的离散傅里叶变换 (DFT)。Y 与 X 的大小相同。如果 X 是向量，则 fft(X) 返回该向量的傅里叶变换; 如果 X 是矩阵，则 fft(X) 将 X 的各列视为向量，并返回每列的傅里叶变换; 如果 X 是一个多维数组，则 fft(X) 将沿大小不等于 1 的第一个数组维度的值视为向量，并返回每个向量的傅里叶变换。

  Y = fft(X,n) 返回 n 点 DFT。如果 X 是向量且 X 的长度小于 n，则为 X 补上尾零以达到长度 n; 如果 X 是向量且 X 的长度大于 n，则对 X 进行截断以达到长度 n; 如果 X 是矩阵，则每列的处理与在向量情况下相同; 如果 X 为多维数组，则大小不等于 1 的第一个数组维度的处理与在向量情况下相同。

  Y = fft(X,n,dim) 返回沿维度 dim 的傅里叶变换。例如，如果 X 是矩阵，则 fft(X,n,2) 返回每行的 n 点傅里叶变换。

- FFT

  在 Matlab 中，FFT（快速傅里叶变换）函数用于将时域信号转换为频域信号。FFT 的基本原理是将一个信号分解成多个正弦和余弦波（即正弦余弦变换），得到每个频率成分的幅值和相位信息。

  fft 之后信号幅度值变大，要得到真实幅度值大小，要将变换后结果除以 N 再乘以 2（直流即零频率处除以 N），除以 N 得到双边频谱，乘以 2 得到单边频谱。如果原始信号就是复数信号，则 fft 得到的就是单边频谱不用乘以 2。

  对 fs 采样频率信号，fft 之后最高频率为采样频率一半，fft 之后的值关于半采样率共轭对称。

  对于大多数 n 值，实数输入的 DFT 需要的计算时间大致是复数输入的 DFT 计算时间的一半。但是，当 n 有较大的质因数时，速度很少有差别或没有差别。

4 实验结果

一共生成了 3 张图，第一张图为双声道原音频，第二张图为快速傅里叶变换得到的音频，第三张图为还原的双声道音频。

快速傅里叶变换进行处理后，由第二张图可知语音信号的能量大部分集中在 20kHz 至 40kHz 之间

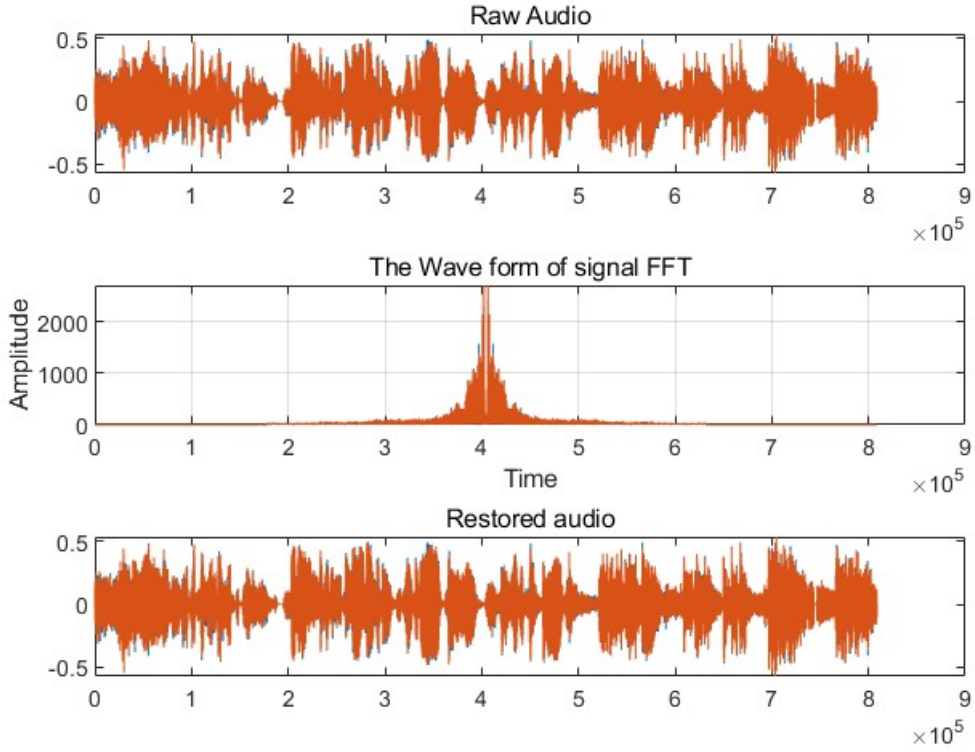


图 4.2: FFT.jpg

5 DWT 离散小波变换

5.1 实验原理

早在 1807 年，法国数学家兼物理学家傅里叶（JeanBaptistle Joseph Fourier）就提出任意一个周期为 $T(2\pi)$ 的函数 $f(t)$ 都可以表示成三角级数, 这就使得复杂的周期函数问题可以简化为三角级数问题。

$$f(t) = \sum_{n=1}^{\infty} C_n e^{int} = \frac{a_0}{2} + \sum_{n=1}^{\infty} a_n cosnt + \sum_{n=1}^{\infty} b_n sinnt$$

其中 $C_n = \frac{1}{2\pi}\int_0^{2\pi} f(t)e^{-int}dt$ ， $a_n = C_n + C_{-n}$ ， $b_n = i(C_n - C_{-n})$, $a_0 \in R$。
将傅里叶级数通过欧拉公式修改为负数形式,

$$f(x) = \sum_{n=-\infty}^{\infty} c_n e^{-i\frac{2\pi nx}{T}}$$

其中 $c_n = \frac{1}{T}\int_{x_0}^{x_0+T} f(x)e^{-\frac{2\pi nx}{T}}dx$
当 T-> ∞，将上式的频率替换为 ω，得到如下形式，其逆过程成为逆傅里叶变换

$$F(\omega) = \frac{1}{2\pi}\int_{-\infty}^{\infty} f(x)e^{-i\omega t}dx$$

$$f(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} F(\omega)e^{i\omega t}d\omega$$

根据傅里叶级数和线性代数相关理论,傅里叶级数的实质可理解为是将信号表示成以 $< 1, sinnt, cosnt >$ 为向量基的向量，且当 $n- > \infty$ 时 $sinnt$ 和 $cosnt$ 均不为 0，这就使得傅里叶变换具有一定的局限性。在特定的表示周期内，若高频信号和低频信号差距很大，高频信号衰减了而低频信号尚未衰减，傅里叶变换的频域表达就会产生误差；再者，三个向量基都是连续的，不能反应振幅的变化情况，傅里叶变换也因此无法体现信号的突变。

对非平稳过程，傅里叶变换有局限性。它只能获取一段信号总体上包含哪些频率的成分，但是对各成分出现的时刻并无所知。因此时域相差很大的两个信号，可能频谱图一样。

5.1.1 STFT 短时傅里叶变换

一个简单可行的方法是加窗。“把整个时域过程分解成无数个等长的小过程，每个小过程近似平稳，再傅里叶变换，就知道在哪个时间点上出现了什么频率了。”这就是短时傅里叶变换 (Short-time Fourier Transform, STFT)。

使用 STFT 存在一个问题，就是应该用多宽的窗函数。窗太窄，窗内的信号太短，会导致频率分析不够精准，频率分辨率差。窄窗口时间分辨率高、频率分辨率低，宽窗口时间分辨率低、频率分辨率高。对于时变的非稳态信号，高频适合小窗口，低频适合大窗口。然而 STFT 的窗口是固定的，在一次 STFT 中宽度不会变化，所以 STFT 还是无法满足非稳态信号变化的频率的需求。

5.1.2 小波分析

所谓小波 $\varphi_{a,b}(t)$ 是由满足条件 $\int_{-\infty}^{\infty}\varphi(\frac{t-b}{a}) = 0$ 和 $C_\varphi = \int_{-\infty}^{\infty}\frac{|\Phi(\omega)|^2}{\omega} < \infty$ 的基小波 $\varphi(t)$ 经过平移和缩放得到的函数族,

$\varphi_(a,b) = \frac{1}{\sqrt{|a|}}\psi(\frac{t-b}{a}) a,b \in R; a \neq 0$

其重构公式 (逆变换) 为:

$f(t) = \frac{1}{C_\psi}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\frac{1}{a^2}\varphi_(a,b)\psi(\frac{t-b}{a})dadb$

其中 a 为尺度因子，用于控制尺度伸缩；b 为平移因子，用于实现小波函数在时间轴上平行移动。根据定义可以看出，小波基能量有限且具有波动特性，因此小波基能够在有限时间内保持特定形状，在有限时间外快速衰减；再者，小波基在周期区间外定义为 0。

离散小波变换需要将连续小波变换中的尺度参数 a 和平移参数 b 进行离散化。因此可以得到相应的离散小波变换为:

$W_f(a,b) =< f, \psi_{a,b}(t) >$

其重构公式为:

$f(t) = \sum_{a,b\in Z} < f, \psi a, b > \psi a, b(t)$

5.1.3 DWT 离散小波变换

小波变换的出发点和 STFT 是不同的。STFT 是给信号加窗，分段做 FFT；而小波直接把傅里叶变换的基给换了——将无限长的三角函数基换成了有限长的会衰减的小波基。这样不仅能够获取频率，还可以定位到时间。

DWT 方法的主要步骤包括:

1. 定义小波基，选择合适的小波基函数;

2. 对信号进行多层分解，将信号分解成多个分量；

3. 对每个分量进行细节系数和近似系数的计算，得到一组小波分量和细节分量；

4. 重构原始信号。

DWT 方法在信号分析和处理方面有广泛的应用，可以有效地去除噪声和提取信号的局部特征。然而，由于其不具有平移不变性，因此在处理某些非平稳和非线性信号时仍存在一定的局限性。

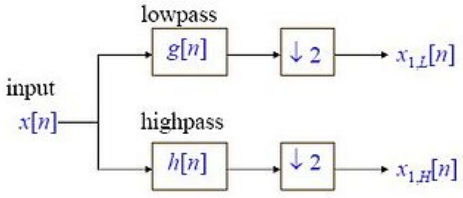
下面定义一些需要用到的信号及滤波器。

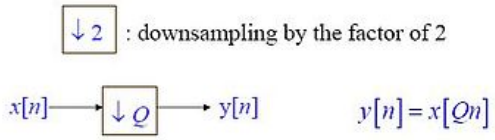
x[n]：离散的输入信号，长度为 N。

g[n]：low pass filter 低通滤波器，可以将输入信号的高频部份滤掉而输出低频部份。

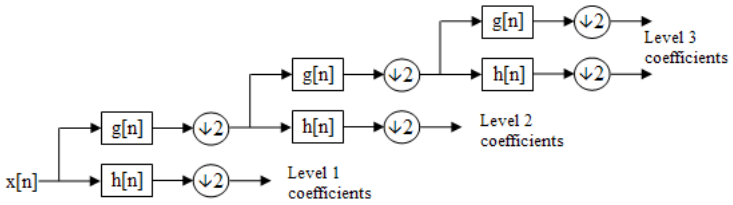
h[n]：high pass filter 高通滤波器，与低通滤波器相反，滤掉低频部份而输出高频部份。

$\downarrow Q$：downsampling filter 降采样滤波器，如果以 x[n] 作为输入，则输出 y[n]=x[Qn]。此处举例 Q=2。





清楚规定以上符号之后，便可以利用阶层架构来介绍如何将一个离散信号作离散小波变换



架构中的第 α 层

$$x_{\alpha,L}[n] = \sum_{k=0}^{K-1} x_{\alpha-1,L}[2n-k]g[k]$$

$$x_{\alpha,H}[n] = \sum_{k=0}^{K-1} x_{\alpha-1,L}[2n-k]h[k]$$

5.2 Matlab 源码

对于离散小波分解，采用了以下几种处理方法

5.2.1 DWT 一级小波分解

利用 dwt() 函数经‘db4’小波进行分解，之后利用 idwt() 函数重构

Listing 3: 一级小波分解

```
%db4一级小波分解与重构DWT
[a,fs]=audioread('testaudio.wav');
[ca1,cd1]=dwt(a(:,1),'db4');%db4一级小波分解，取原音频的
a0=idwt(ca1,cd1,'db4',length(a(:,1)));%db4一级小波分解重构

figure('name','一级小波分解与重构DWT');
ax(1)=subplot(4,1,1);
plot(a(:,1));%原始波形
title('Raw wave')
grid on;

ax(2)=subplot(4,1,2);
plot(cd1);%细节分量
title('Component of Detail')
grid on;

ax(3)=subplot(4,1,3);
plot(ca1);%近似分量
title('Approximate Component')
grid on;

ax(4)=subplot(4,1,4);
plot(a0);%一级分解重构的结果
title('Recovered wave')
grid on;

linkaxes(ax,'x');
```

5.2.2 wavedec 一级小波分解

利用 wavedec() 函数经‘db4’小波进行分解，之后分解重构

Listing 4: 一级小波分解

```
[a,fs]=audioread ("testaudio.wav") ;
[ca1,cd1]=wavedec(a(:,1),1 ,'db4') ;
a0=waverec(ca1,cd1,'db4') ;
%绘图
subplot (2 ,2 ,1) ; plot ( a ( : , 1)) ;
subplot (2 ,2 ,2) ; plot ( cd1 ) ; %细节分量
subplot (2 ,2 ,3) ; plot ( ca1 ) ; %近似分量
subplot (2 ,2 ,4) ; plot ( a0 ) ;
axes_handle = get ( gcf,'children') ;
axes ( axes_handle (4) ) ; title('Raw wave') ;
axes ( axes_handle (3) ) ; title('Detail Component') ;
axes ( axes_handle (2) ) ; title('Approximate Component') ;
```

```
axes ( axes_handle (1) ) ; title('Recovered wave') ;
```

dwt2 是二维单尺度小波变换，其可以通过指定小波或者分解滤波器进行二维单尺度小波分解。而 wavedec2 是二维多尺度小波分解.

尺度可理解为级，即 wavedec2 可用于多级小波分解

dwt2: [cA,cH,cV,cD]=dwt2(X,'wname');

wavedec2: [C,S]=wavedec2(X,N,'wname'), 其中 N 为大于 1 的正整数。

dwt2 只能对某个输入矩阵 X 进行一次分解，而 wavedec2 可以对输入矩阵 X 进行 N 次分解。

5.2.3 wavedec 三级小波分解

利用 wavedec() 函数经‘db4’小波进行分解，同时给出原始语音信号、三级分解的细节分量、一级分解的近似分量、二级分解的近似分量、三级分解的近似分量，最后是三级分解重构的结果。

Listing 5: 三级小波分解

```

%db4小波三级分解与重构
[a,fs]=audioread('testaudio.wav');
[c,l]=wavedec(a(:,2),3,'db4');%db4小波三级分解，取原音频的另一个声道

ca3=appcoef(c,l,'db4',3);%第三级近似分量，低频
cd3=detcoef(c,l,3);%第三级细节分量，高频
cd2=detcoef(c,l,2);%第二级细节分量，高频
cd1=detcoef(c,l,1);%第一级细节分量，高频
a0=waverec(c,l,'db4');%db4小波三级重构

figure('name','三级小波分解与重构');
ax(1)=subplot(6,1,1);plot(a(:,2));grid on;
ax(2)=subplot(6,1,2);plot(cd1);grid on;
ax(3)=subplot(6,1,3);plot(cd2);grid on;
ax(4)=subplot(6,1,4);plot(cd3);grid on;
ax(5)=subplot(6,1,5);plot(ca3);grid on;
ax(6)=subplot(6,1,6);plot(a0);grid on;
linkaxes(ax,'x');
```

5.3 实验结果

5.3.1 DWT 一级小波分解

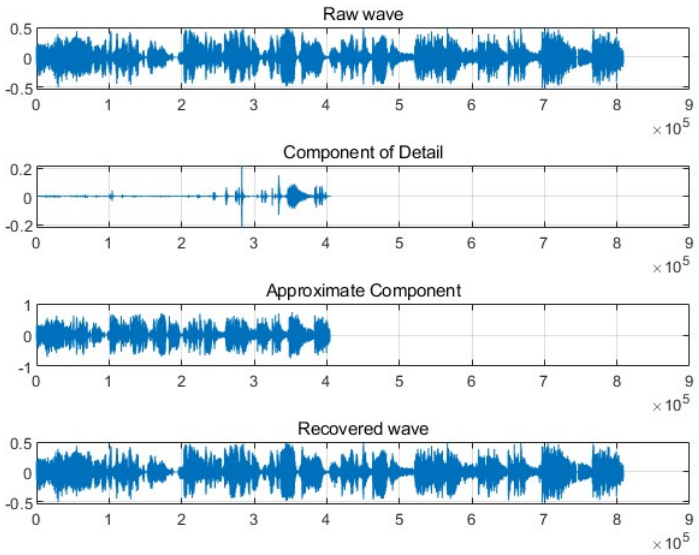


图 5.3: DWT 一级小波分解.jpg

取其中一个声道的音频做一级小波变换。第一张图为原音频语音信号; 第二张图为一级小波分解的细节分量，也就是高频部分，数据长度较原数据缩短一半；第三张图为一级小波分解的近似分量，也就是低频部分，数据长度较原数据缩短一半；第四张图为分解重构的结果。

5.3.2 wavedec 一级小波分解

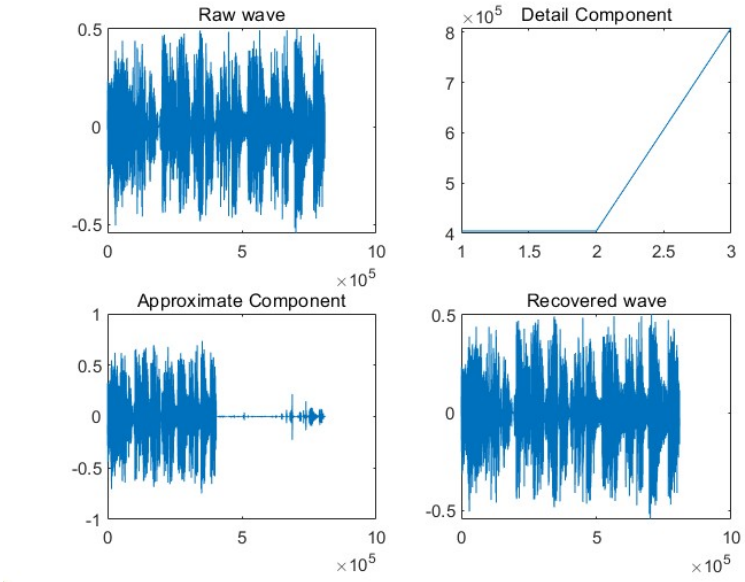


图 5.4: wavedec 一级小波分解.jpg

取其中一个声道的音频做一级小波变换。第一张图为原音频语音信号; 第二张图为一级小波分解的细节分量，也就是高频部分，数据长度较原数据缩短一半；第三张图为一级小波分解的近似分量，也就是低频部分，数据长度较原数据缩短一半；第四张图为分解重构的结果。

5.3.3 wavedec 三级小波分解

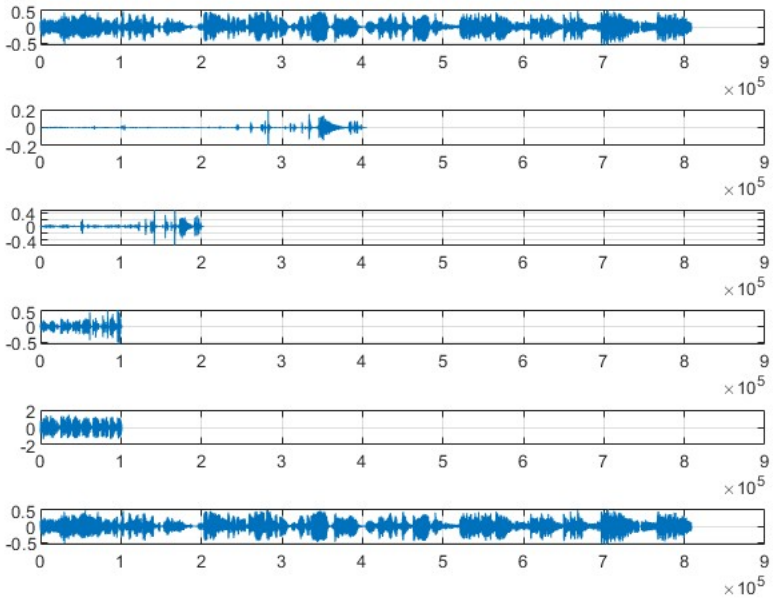


图 5.5: wavedec 三级小波分解.jpg

取一个声道的音频做三级小波变换。第一张图为原音频语音信号；第二张图为一级小波分解的细节分量，也就是高频部分，数据长度较原数据缩短一半。第三张图为二级小波分解的细节分量，也就是高频部分，数据长度较一级数据缩短一半；第四张图为三级小波分解的细节分量，也就是高频部分，数据长度较二级数据缩短一半；第五张图为二级小波分解的近似分量，也就是低频部分，数据长度较二级数据缩短一半；第六张图为分解重构的结果。

6 DCT 离散余弦变换

6.1 实验原理

傅里叶变换的形式有很多种，归一化的二维离散傅里叶变换（Discrete Fourier transform, DFT）可以写成如下形式：

$$F(u,v) = \frac{1}{\sqrt{NM}} \sum_{x=0}^{N-1} \sum_{y=0}^{M-1} f(x,y) e^{-\frac{2\pi i}{N}ux} e^{-\frac{2\pi i}{M}vy}$$

$$f(x,y) = \frac{1}{\sqrt{NM}} \sum_{u=0}^{N-1} \sum_{v=0}^{M-1} F(u,v) e^{\frac{2\pi i}{N}ux} e^{\frac{2\pi i}{M}vy}$$

傅里叶变换包含复数运算，其运算复杂度和存储长度都超过实数运算。为了简化上述过程，同时达到更好的变换效果，产生了余弦变换。

从傅里叶变换到离散余弦变换，需要一些数学理论的支持。在给定区间内满足狄利赫里条件的连续实对称函数，可以展开成仅含有余弦项的傅里叶级数。

对于定义在正实数域上的函数，可以通过偶延拓或奇延拓，满足上述条件。但如果函数的定义域包含零点，情况则稍有些复杂。

以一个二维离散函数 $f(x,y)(x,y=0,1,...,N-1)$ 为例，对其进行偶延拓。

假如序列中不包含零点，按照以下方式延拓：

$f(1,0)=f(-1,0), f(0,1)=f(0,-1)$, 对称中心为 (0,0)

由于序列中包括零点，考虑零点后的延拓方式：

$f(0,0)=f(-1,0), f(0,0)=f(0,-1)$ ，对称中心为 $(-\frac{1}{2},-\frac{1}{2})$

因此，按照上述方法延拓后，归一化的二维离散余弦变换可以写成如下形式：

$$when\,(x,y)\,or\,(u,v)=(0,0)$$

$$F(u,v)=\frac{1}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}f(x,y)cos[\frac{\pi}{N}u(x+\frac{1}{2})]cos[\frac{\pi}{N}v(y+\frac{1}{2})]$$

$$f(x,y)=\frac{1}{N}\sum_{u=0}^{N-1}\sum_{v=0}^{N-1}F(u,v)cos[\frac{\pi}{N}u(x+\frac{1}{2})]cos[\frac{\pi}{N}v(y+\frac{1}{2})]$$

$$when\,(x,y)\,or\,(u,v)\neq(0,0)$$

$$F(u,v)=\frac{1}{2N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}f(x,y)cos[\frac{\pi}{N}u(x+\frac{1}{2})]cos[\frac{\pi}{N}v(y+\frac{1}{2})]$$

$$f(x,y)=\frac{1}{2N}\sum_{u=0}^{N-1}\sum_{v=0}^{N-1}F(u,v)cos[\frac{\pi}{N}u(x+\frac{1}{2})]cos[\frac{\pi}{N}v(y+\frac{1}{2})]$$

在傅里叶变换中，正逆变换的变换核 $e^{-\frac{2\pi i}{N}ux}e^{-\frac{2\pi i}{M}vy}$ 与 $e^{\frac{2\pi i}{N}ux}e^{\frac{2\pi i}{M}vy}$ 相差一个负号；

相应的，在余弦变换中，正逆变换的变换核 $cos[\frac{\pi}{N}u(x+\frac{1}{2})]cos[\frac{\pi}{N}v(y+\frac{1}{2})]$ 也应相差一个负号。由于 $cos(x)=cos(-x)$ ，所以余弦变换的正逆变换在形式上具有一致性。

6.2 Matlab 源码

利用 dct() 函数经‘db4’小波进行分解，之后利用 idct() 函数重构

Listing 6: DCT 离散余弦变换

```
1
2  %离散余弦变换
3  [a,fs]=audioread('testaudio.wav');
4  da=dct(a);%离散余弦变换
5  a0=idct(da);%离散余弦逆变换4
6
7  figure('name','离散余弦变换');
8  subplot(3,1,1);plot(a);%原始波形
9  title('Raw wave')
10
11 subplot(3,1,2);plot(da);%dct()处理后
```

```
12 title('DCT')
13
14 subplot(3,1,3);plot(a0);%idct()重构后
15 title('Restored wave')
```

6.2.1 Matlab 中的 DCT 函数

- 语法

  y = dct(x)

  y = dct(x,n)

  y = dct(x,n,dim)

  y = dct(___,'Type',dcttype)

- 说明

  离散余弦变换 (DCT) 与离散傅立叶变换密切相关。通常可以从几个 DCT 系数非常精确地重建序列。此属性对需要数据减少的应用程序很有用。

  DCT 有四种标准型号。对于长度为 N 的信号 x，以及具有 δkl 的 Kronecker delta，变换由下式定义

  - DCT-1:

$$y(k)=\sqrt{\frac{2}{N}}\sum_{n=1}^{N}x(n)\frac{1}{\sqrt{1+\delta_{n1}+\delta_{nN}}}\frac{1}{\sqrt{1+\delta_{k1}+\delta_{kN}}}\cos\left(\frac{\pi}{N-1}(n-1)(k-1)\right)$$

  - DCT-2:

$$y(k)=\sqrt{\frac{2}{N}}\sum_{n=1}^{N}x(n)\frac{1}{\sqrt{1+\delta_{k1}}}\cos\left(\frac{\pi}{2N}(2n-1)(k-1)\right)$$

  - DCT-3:

$$y(k)=\sqrt{\frac{2}{N}}\sum_{n=1}^{N}x(n)\frac{1}{\sqrt{1+\delta_{n1}}}\cos\left(\frac{\pi}{2N}(n-1)(2k-1)\right)$$

  - DCT-4:

$$y(k)=\sqrt{\frac{2}{N}}\sum_{n=1}^{N}x(n)\cos\left(\frac{\pi}{4N}(2n-1)(2k-1)\right)$$

  该系列从 n = 1 和 k = 1 索引，而不是通常的 n = 0 和 k = 0，因为 MATLAB 向量从 1 到 N 而不是从 0 到 N - 1。

  DCT 的所有变体都是单一的 (或等效地，正交)：要找到它们的反转，在每个定义中切换 k 和 n。特别地，DCT-1 和 DCT-4 是它们自己的逆，并且 DCT-2 和 DCT-3 是彼此相反的。

6.3 实验结果

第一张图为原始音频语音信号，第二张图为离散余弦变换得到的音频，第三张图为重构后的音频语音信号

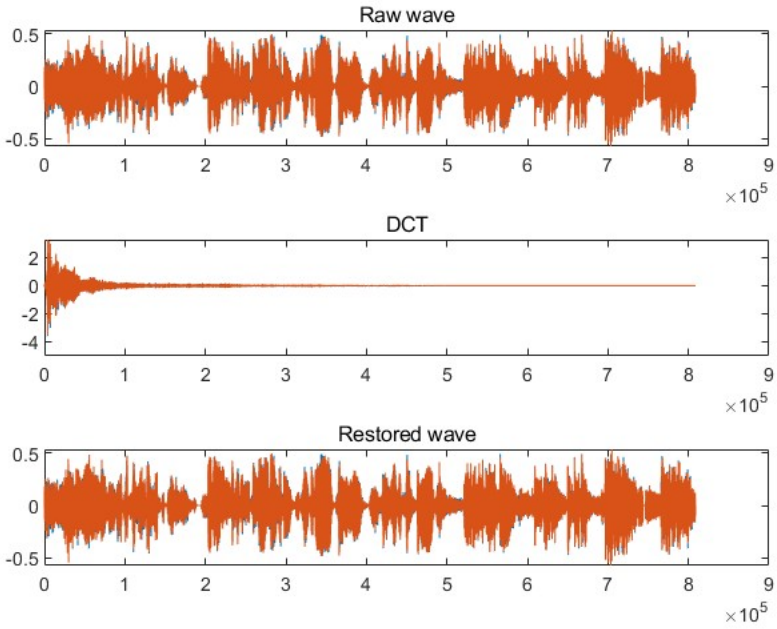


图 6.6: DCT 离散余弦变换.jpg

参考文献

[1] https://zhuanlan.zhihu.com/p/347091298

[2] https://zhuanlan.zhihu.com/p/22450818

[3] https://blog.csdn.net/weixin_39852875/article/details/102727474

[4] https://zhuanlan.zhihu.com/p/85299446

[5] https://zhuanlan.zhihu.com/p/33845296