基于变换的无损音频压缩算法设计

摘 要

无损音频压缩是指在不损失任何音频信息的前提下实现压缩。目前的有损编码标准虽能提供良好的主观音质和高压缩比，但对于动态范围较大的音乐、具有高质量要求的演播室或者音乐厅环境中、以及具有二次编码要求的音频编辑应用中，有损编码无法满足质量要求。而无损编码能达到真正的透明音质，且在不同的无损格式之间相互转化不丢失任何音频信息，因而研究具有高压缩比的无损音频压缩算法具有重要的理论意义和实用价值。

本文设计了基于整数改进离散余弦变换（IntMDCT）的无损音频压缩算法，输入音频数据首先进行分帧加窗，然后进行IntMDCT，变换后的系数经映射后进行Golomb-Rice熵编码，同时论文设计了相应的比特流结构，实验结果表明本文提出的无损音频压缩算法具有良好的压缩性能。

改进型离散余弦变换（MDCT)技术在音频压缩中广泛使用，但由于传统MDCT基于浮点算法设计，在定点实现时由于存在截断误差而无法实现完全可逆。因此本论文研究整数MDCT（IntMDCT）以实现真正可逆变换并将之用于无损音频压缩。论文证明了MDCT分解为Givens旋转和DCT的推导过程，在此基础上使用两种提升方案——“经典提升”和“多维提升”实现IntMDCT。同时论文对IntMDCT

中复杂的矩阵运算提出了相应简化算法，降低了算法实现的复杂度。

为了便于编码，本文对变换后的系数进行了映射处理，并在大量统计分析的基础上，发现经过映射后的IntMDCT系数的概率分布接近于几何分布，因此算法中采用了Golomb-Rice熵编码技术，并且达到较好的编码性能。由于不同频段内变换系数的幅度差异，本论文对于不同频段的变换系数采用了不同的Golomb-Rice编码参数以得到最优的压缩效率。

论文同时设计了适合于该无损音频压缩算法的比特流结构，并实现了该无损音频编解码算法，分析了采用两种提升实现IntMDCT相对于浮点变换结果的逼近误差，以及对于整个算法压缩性能的影响。论文最后对算法的压缩效率、鲁棒性和复杂度等性能进行分析。对SQAM中70个测试曲目编码的实验结果表明，本算法的平均压缩比为3.17，具有较高的压缩效率；并且本算法具有较好的鲁棒性和较低的复杂度。

关键词：无损音频压缩，整数变换，整数改进离散余弦变换，熵编码

RESEARCH OF LOSSLESS AUDIO COMPRESSION BASED ON ORTHOGONAL TRANSFORM

ABSTRACT

Lossless audio compression means to compress audio signals without any information loss. At present, lossy audio compression algorithms can provide good subjective quality and high compression ratio. However, they can not meet the requirements in such scenarios as high quality studio or theater applications, large-dynamic-range music coding and audio editing with probability to recoding. Compared with the lossy audio compression, the lossless audio compression can obtain truely transparent audio quality, and will not lose any audio information when transcoding between two lossless format. So to research of lossless audio compression with high compression ratio has a great significance in both theory and practical applications.

This thesis presents a lossless audio compression algorithm based on Integer Modified Discrete Cosine Transform(IntMDCT) and the corresponding bit-stream structure. The proposed compression algorithm first divides the input audio data into frames and multiplies them with

proper window function, then carry through IntMDCT, and finally codes the mapped transform coefficients with Golomb-Rice code. The simulation results show that the proposed lossless audio compression algorithm has a good compression performance.

MDCT (Modified Discrete Cosine Transform) is widely used in the field of audio compression. Originally designed for floating-point algorithms, the MDCT is usually not invertible when implemented on fixed point processor. So this thesis researches into the invertible Integer MDCT (IntMDCT), which produces a mapping mechanism between an integer sequence and the integer transform coefficients. The mathematical deduction process from the MDCT to the Givens rotation and DCT is also presented, together with the implementation of the integer MDCT by the two lifting schemes – classical lifting and multi-dimensional lifting. The thesis gives reduced methods of some complex matrix decomposition involved within the implementation of integer MDCT.

Golomb-Rice code is a special kind of Huffman code suitable for the coding of positive integer signals of geometric distribution. This thesis finds that the probability distribution of the mapped transform coefficients is close to geometric distribution through a statistical analysis of a large number of audio signals. Therefore the algorithm can achieve high coding gain using Golomb-Rice code. Besides, the algorithm uses different Golomb-Rice code parameter in different frequency range to

achieve high compression ratio in terms of the corresponding magnitude difference.

The thesis also designs a suitable bit stream structure for the proposed lossless audio compression algorithm and implements the porposed algorithm. The thesis analyzes the approximating error margins of the two lifting schemes to realize the IntMDCT to their floating point counterpart, and their influence on the compression efficiency of the whole algorithm. Finally the thesis analyzes the compression ratio, robustness and computational complexity of the algorithm. Experimental results of compressing 70 test tracks in SQAM show that the average compression ratio of the proposed algorithm is about 3.17, and it also has good robustness and low computational complexity.

KEY WORDS: lossless audio compression，integer transform，integer modified discrete cosine transform (IntMDCT)，entropy coding

上海交通大学

学位论文原创性声明

本人郑重声明：所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的作品成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

学位论文作者签名：朱敏

日期：2006 年 1 月 17 日

上海交通大学
学位论文版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，同意学校保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权上海交通大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

保密□，在___年解密后适用本授权书。

本学位论文属于

不保密√。

（请在以上方框内打“√”）

学位论文作者签名：朱敏　　指导教师签名：胡剑凌

日期：2006 年 1 月 17 日　　日期：2006 年 1 月 17 日

第一章 绪论

数字化音频带给人们更多方便的同时也由于其数据的海量性，给信息的存储和传输造成较大困难，成为阻碍人类有效获取和使用多媒体信息的瓶颈问题之一。解决这一问题的办法，单纯从扩大存储器容量、增加通信干线的传输速率考虑是不现实的，而对数据信息进行压缩，以压缩编码形式存储和传输，在解码端和接收端再对数据解压缩恢复原来信息则是一个行之有效的方法。事实证明，多媒体数据压缩不仅是必要的而且是可行的，原因是声音、图像等信源数据有较强的相关性，即存在大量的冗余信息。可通过去掉冗余信息（去除数据间的相关性）、保留独立信息分量来实现压缩。因此，研究开发高效的音频编码方法，以压缩的形式存储和传输音频信息是很好的选择。随着人们对音频质量要求的提高，如何在保留全部音频信息的条件下，以尽可能大的压缩比压缩音频，从而给人们提供真正透明的音质，成为当前无损音频压缩所面临的主要课题。

1.1 课题背景与意义

从 1971 年英国 BBC 将数字技术用于声音录制开始到现在，数字音频技术已经发展了三十多年，其中具有划时代意义的是八十年代初出现的 CD 和数字音频磁带（DAT），它们在诞生之际便向大众展示了数字音频的优点，高保真的质量和高稳定度，但这些优点都建立在大容量的数据存储基础上，传统的 CD 和 DAT 录制的音频信号是经过 44.1 或 48kHz 采样，每个采样点用 16 比特进行 PCM 编码。这样，单声道每秒要 705.6/768 Kbit ，立体声每秒要 1.41/1.54 Mbit。虽然数据量如此高，但作为第一代数字音频产品的 CD 和 DAT 主要面向存储，所以依然采纳了这种技术。紧接着出现的第二代多媒体产品，特别是无线系统，受限于带宽和产品成本，无法支持高速率的数据。因此，为开辟新的无线广播和网络多媒体应用，要求在接近透明声音质量的情况下压缩数据量，即编码后重建的声音质量与原始声音对人耳感知上具有不可区分的特性。正是由于这种需求的存在，推动了有损音频压缩技术的迅猛发展。但是由于有损编码后的音

质与原声 CD 存在区别，并随着压缩码率的降低质量下降，这对追求完美的听众来说难以接受，更无法应用于专业领域和存档系统。现在，随着大容量存储设备的研制、高速宽带网的建设，大大推动了无损音频压缩算法的研究和应用。

音频编码从信息是否有所损失的角度可以分为两类：有损音频压缩和无损音频压缩。有损音频压缩也称信息量压缩，这种压缩方法利用了人类听觉对声音中的某些频率成分不敏感的特性，从原始数据中将这一部分人类听觉不敏感的数据去除，以达到压缩的目的。所损失的部分对聆听原始声音的影响较小，却换来较大的压缩比，但是不能完全恢复原始数据是有损压缩方法最大的缺点。顾名思义，无损音频压缩是指在不损失任何音频信息的前提下实现压缩。无损压缩也称冗余度压缩，它利用数据的统计冗余进行压缩，这种压缩方法从数学上讲是一种可逆运算，还原后和压缩编码前的数据完全相同。不存在数据丢失的问题是无损压缩最大的优点。

目前的有损编码标准（如 MP3 和 AC-3）在大多数情况下能够达到良好的主观音质和高压缩比，但遇到数据动态范围较大的音乐（如交响乐等）时，其音质就差强人意了。在一些演播室或者音乐厅环境中，有损编码后的音频效果无法达到期望的逼真度。另外，在音频编辑的过程中经常需要对音频数据作二次编码（即从一种有损格式转换成另一种有损格式，或者格式不变仅改变比特率），如 VCD 音频压缩格式为 mp2(MPEG-1 layerII)，需转化为 mp3(MPEG-1 layerIII)格式用于因特网上传输。不同压缩格式的码流之间无法直接转化，只能先对 mp2 解码为 pcm 格式，然后对 pcm 进行 mp3 编码。由于有损编码丢失了一部分的音频数据，有损压缩格式的二次编码意味着将丢失更多的信息，引入更大的失真。与有损压缩相比，无损压缩后能达到真正透明音质，且在不同的无损格式之间相互转化不丢失任何音频信息。

无损压缩技术可以给人们带来完美音质的听觉享受。以前，由于声卡等电脑多媒体设备的落后，硬盘容量以及网络带宽的限制，人们更多注重于数字音乐编码技术的压缩比，力求在可以接受的品质下将数字音乐文件压缩得尽量小。但随着多媒体设备的发展以及宽带网络建设与大容量硬盘的普及，大部分音乐爱好者对数字音乐文件的期望已经不再是文件尽量小，而是追求所谓的“CD 音质”播放效果。对于希望通过互联网传输 CD 音质的数字音乐文件的用户来说，

无损音频压缩无疑是非常有吸引力的一种方法。一种合理的建议是，提供经过有损编码后的压缩的音频给消费者用于网上点播，并且可以提供给喜欢音频 CD 音质的消费者一个经过无损音频编码后的音频版本下载收藏。

无损音频技术可以用于专业领域如录音室、演播室和存档系统。在音频编辑过程中如需要对音频信号进行修改或者是不同编码格式之间转换时，无损压缩避免了由于使用有损压缩编码器的多级编解码而导致的音频降质。

无损音频编码还应用于存储高分辨率和高采样率音频信号的新 DVD 标准中。

无损音频编码也可以用于语音增强、语音识别、说话人识别等需要检测语音特征的应用场合中原始语音的压缩。

无损音频压缩唯一的不足是目前算法得到的压缩比仍不够高，限制了无损音频压缩的广泛应用。随着大容量的存储设备的研制、高速宽带网的建设以及各种无损音频压缩算法的研究，将大大推动无损音频压缩的应用。可以预期，未来应用有损音频压缩的场合都可以使用无损音频压缩代替，如：汽车、家庭影院、消费类音频和个人计算机（PC）等等，从而给人们带来更为完美的听觉享受。

1.2 无损音频压缩的现状

与有损音频压缩相比，无损音频编码方面的所作研究相对较少。一些著名的时域无损音频编码的方法有 Monkeys Audio，Shorten，LTAC, MusiCompress 和 AudioPak 等算法。总的来看，这些无损音频压缩方法可以分为两类，基于预测的和基于变换的方法。预测编码一般是使用线性预测或者自适应预测器，变换编码一般使用整数 DCT、整数 MDCT 和整数小波变换来去除信号的相关性。表 1—1 中列出当前各种无损音频算法，相信它们代表了无损音频压缩算法的现状。

Monkeys Audio[32]：现有无损音频算法中，Monkeys 音频是对于大部分曲目有着最高压缩比的算法，其编码后后缀名为.APE，现有大部分的播放器可以直接或者安装插件来播放这种格式的文件。对于联合立体声采用 Mid/Side（M/S）编码，将左右声道的信息转化为左右声道的中点（mid）和左右声道之差（side）

来编码。采用自适应预测器去除音频样点的相关性，使用 Rice 码编码预测系数和残差。

FLAC[1] (Free Lossless Audio Codec)：此算法是公开源码，人们都可以通过互联网免费得到源代码，并在此基础上作改进，便于人们交流对于无损音频压缩算法的研究。FLAC 编码器分为以下几个步骤：分块、信道内去相关、预测、残差编码。具体参见 FLAC 的网页[1]。

表 1—1 现有著名无损音频算法列表

Table 1-1 List of Lossless Compression Algorithms

算法名称	作者
Monkeys Audio [32]	Matthew T. Ashland
FLAC[1]	Josh Coalson
LPAC[33]	Tilman Liebchen
LTAC[25]	M. Purat, T. Liebchen, P. Noll
Shorten[24]	T. Robinson
AudioPak[23]	M. Hans, R. Schafer
AudioZip[21]	Lin Xiao
Waveform Archiver(WavArc)[29]	D. Lee
OggSquish[30]	C. Montgomery
Sonarc[31]	R. Sprague
MUSICompress[2]	A. Wegener
DVD[27][28]	P. Craven et al.
Philips[3]	A. Bruekers, A. Oomen, R. van der Vleuten

LTAC[25]（Lossless Transform Audio Codec, DOS 下版本 1.01）和 LPAC[33]（Lossless Prediction Audio Codec）均为 Tilman Liebchen 等人所研究，LTAC 是唯一使用变换编码去相关的编解码器。它使用离散余弦变换（DCT）去相关，Rice 编码用于打包变换系数和残差。LPAC (Lossless Predict audio coding)编码器则是采用自适应预测器和熵编码。

Shorten[24]（DOS 下版本 2.1）：提出时域的线性预测方法。使用两种命令：

-p 0: shorten –b 1152 –v 2 –t s16 –p 0 file.pcm filep0.shn

和

-p 10: shorten –b 1152 –v 2 –t s16 –p 10 file.pcm filep0.shn

这两种命令定义了不同的编码器，不同之处在于信道内去相关模块。选项-p 0 表示对于整数系数使用 FIR 预测器，选项-p 10 表示使用十阶最小二乘线性 FIR 预测器。这两种编码器均使用 Rice 编码预测残差信号。

Waveform Archiver[29] (WavArc DOS 下版本 1.1)、Sonarc[31] (DOS 下版本 2.1b)

和 OggSquish[30]（Unix 下版本 98.9）：这三种算法都是来自于个人通信领域的算法设计者。Waveform Archiver 使用 FIR 线性预测器去除信道内的相关性，熵编码模块采用 Rice 编码。Sonarc 使用 FIR 线性滤波器和哈夫曼编码。版本号为 98.9 的 OggSquish 算法的无损音频编码器使用 IIR 线性预测和哈夫曼编码。

AudioZip[21]是新加坡南洋技术大学信号处理中心 LinXiao 提出，使用线性预测 LPC（linear predict coding）去相关，使用 Rice 码编码预测系数和残差。详细可参考文献[21]。

AudioPaK[23]是一种低复杂度的音频无损压缩算法，包含预测和熵编码两个模块。其中预测器是在四个整数系数的滤波器中选一个最佳滤波器；熵编码则采用 Golomb-Rice 编码；未采用联合立体声编码去除信道间的相关性。从算法复杂度低而效率又较高这个角度来讲，该编码器是相当不错的。

MUSICompress[2]（Windows 下版本 1.2）：MUSICompress 使用预测型计算结构，从原始信号中减去根据原始信号自适应变化的的近似值来产生无损编码的误差信号。MUSICompress 使用浮点表示和哈夫曼码编码近似值和残差信号。

DVD 标准：这个算法基于 Craven et al 的著作。它使用 IIR 预测，对于预测残差信号进行哈夫曼编码。

Philips：算法由 Philips 提出[3]，使用一个十阶 FIR 线性预测器和 Rice 编码残差信号。

1.3 无损音频压缩标准（MPEG-4 ALS）简介

MPEG-4 提供了基于对象的音频编码方法，包括自然语音、通用音频、结构化音频、音频合成等工具，分别为自然音频、合成音频提供编码合成方法。MPEG- 4 通用音频以改进的 AAC 算法为主，应用在除最低码率之外的一般性的音频编码中，并采用 TwinVQ 算法来解决 CELP 语音编码在处理音乐信号时的缺点；HVXC 和 H ILN 是 MPEG - 4 提供的参数化编码工具，HVXC 适用于处理语音信号,速率为 2－4 kbps，H ILN 适用于处理非语音信号，速率大于 4 kbps；MPEG- 4 SA (结构化音频)的核心 SAOL 可以描述声音处理和声音合成算法。由于在档案系统、演播室、宽带传输等许多记录和传输音频信号的应用中，需要采用无损压缩的方式，而 AAC、MP3 等感知音频编码的算法采用有损压缩

的方式，MPEG 音频工作组正在进行无损压缩编码技术的标准化工作。

MPEG-4 ALS标准化的过程及进展如下：

2002年7月，MPEG音频工作组提议提交无损音频压缩技术，并对无损音频压缩方案的兼容性、可分级性以及随机访问能力提出了一些具体要求。这个提议基于两种不同的方法，一种包括有损编码和无损编码的分级系统，另一种只包含独立的无损编码方案的系统。

2002年12月，7家公司分别提交了一种或多种满足基本要求的音频编解码器。

2003年3月，MPEG音频工作组决定首先进行只有无损编码的音频编解码器的标准和工作，同时继续研究分级系统。

2003年7月，柏林工业大学提交的编码方案被确定为工作草案。

2003年10月，将RealNetworks提议的熵编码方案添加到工作草案中。

MPEG4 已在 2004 年底制定了纯无损音频压缩标准（简写为 MPEG-4 ALS），同时对可分级编码方案继续做研究和评估。最终新加坡信息与通信研究院（Institute for infocomm Research）的 AAZ（Advanced Audio Zip）技术被采纳为 MPEG-4 SLS（MPEG-4 Scaleable to Lossless Standard）的参考模型。

对于无损音频压缩，为了编解码一致，需对编码器作部分规定。MPEG-4 ALS 建议的编码器主要模块如图 1—1 所示[22]。

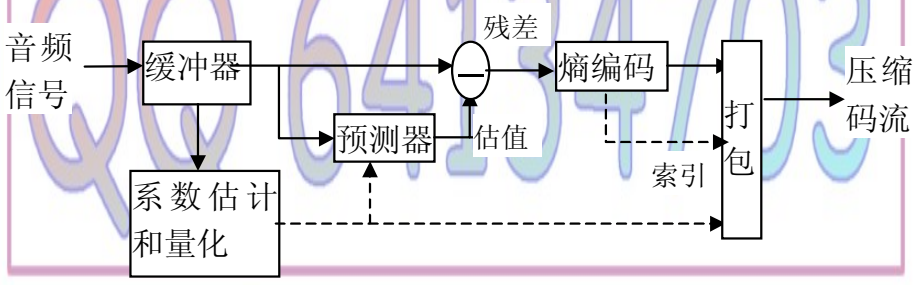


图 1—1　MPEG-4 ALS 编码器结构

Figure 1-1 MPEG-4 ALS encoder

其压缩过程是：首先利用线性预测（LPC）技术得到预测残差信号，以去除信号间的相关性，再对残差信号进行熵编码。其中的预测器采用 Levinson - Durbin 算法，自适应地选择最优阶数和系数，再对系数量化和编码，作为参数放入码流中。对残差的熵编码可动态地在 Golomb - Rice 码和高效块码中选择效率较高

的一种，选择索引也放入码流。

另外标准中还包括三个附加选项：随机存取，即可以从任意帧开始解码或剪辑，因此需要对音频信号分帧，每帧独立处理，帧长与采样率有关，根据音频工程师协会和欧洲广播联合会的推荐，帧长选 192 的整数倍；块长切换，即每帧自适应地决定是否细分成四个子块，分别选择预测和熵编码方案；联合立体声编码，即从左声道、右声道和（左－右）声道中选择效率高的两个进行编码。

1.4 本论文研究内容与主要价值

无损音频压缩从去除信号样值相关性的角度可以分为预测编码和变换编码两种方法。LTAC（Lossless Transform Audio Codec）是上述现有算法中唯一基于变换的算法，它采用离散余弦变换（DCT）去除音频信号的相关性。本课题主要讨论基于整数 MDCT 变换的无损音频压缩算法的设计，在计算复杂度允许的情况下，尽可能得到大的压缩比。

本文的研究内容有以下 4 个方面：

首先，研究基于提升技术和 Givens 旋转的整数变换，特别是整数 DCT 和整数 MDCT 的推导和实现过程。分析整数变换中使用的取整次数造成了整数变换后的结果与浮点变换结果之间的误差，以及该误差对反映信号频谱特征的影响。

第二，研究熵编码技术。分析变换后系数分布的特征，选择合适的编码方法以及编码参数。

第三，研究适合编解码算法的比特流结构设计。

第四，研究无损音频压缩算法的整体架构设计。综合考虑各方面因素，在复杂度允许的情况下，得到较高的压缩比。分析本算法的压缩性能，并与其他现有算法进行比较，指出本算法对于何种类型的音频信号会具有比较好的压缩效率，分析其技术根源。

论文的主要价值在于：

第一， 论文证明了 MDCT 分解为 Givens 旋转和 DCT 的推导过程，在此基础上使用两种提升方案－－“经典提升”和“多维提升”实现 IntMDCT。同时论文对 IntMDCT 中复杂的矩阵运算提出了相应简化算法，降低了算法实现的复

杂度。

第二， 本文对变换后的系数进行了映射处理，并在大量统计分析的基础上，发现经过映射后的 IntMDCT 系数的概率分布接近于几何分布，因此算法中采用了 Golomb-Rice 熵编码技术，并且达到较好的编码性能。由于不同频段内变换系数的幅度差异，本论文对于不同频段的变换系数采用了不同的 Golomb-Rice 编码参数以得到最优的压缩效率。

第三， 本文提出并实现了一种新的无损音频编解码算法，设计了适合于该无损音频压缩算法的比特流结构，实现音频数据的有效编码。分析了采用两种提升实现 IntMDCT 相对于浮点变换结果的逼近误差，以及对于整个算法压缩性能的影响。

第四，论文对本算法的压缩效率、鲁棒性和复杂度等性能进行分析，并与现有无损算法相比较。本算法对于各种音频信号均有较好的压缩效果，特别是对于乐器音频的压缩甚至超过了 APE。

1.5 论文的结构安排

本论文第二章介绍无损音频压缩的原理和关键技术，第三章介绍整数 MDCT 变换的实现，第四章提出基于整数 MDCT 的无损音频压缩算法设计，第五章给出算法实现以及性能分析，第六章是总结和展望。

第二章　无损音频压缩的原理和关键技术

无损压缩，又称为可逆压缩，无失真编码，无噪声编码。无损音频压缩是指在没有任何音频信息损失的前提下，降低数据量，获得低比特率的音频数据表示。Shannon在创立信息论时，把数据看成是信息和冗余度的组合，无损压缩的工作原理是去除数据中的冗余度而不损失任何信息，因此是一个可逆的过程。

2.1 无损压缩的原理

1948年，Shannon发表了《通信的数学原理》，应用概率论对信息作了定量的描述。设信源为X，集合A＝$\{x_1, x_2, ..., x_n\}$为表示信息内容的符号集合，如果各符号 x_i 的出现是各不相干的，或者说是独立的，在信息论中称信源 X 是无记忆的，设 $P(x_i)$ 为 X 发出符号 x_i 这一事件的概率，Shannon定义 x_i 的信息量 $I(x_i)$ 为

$$I(x_i) = -\log P(x_i) \tag{2—1}$$

以2为底时，$I(x_i) = -\log_2 P(x_i)$，信息量的单位为比特（bit）。

可见，一个符号的出现概率越小，其包含的信息量就越多；相反地，一个符号的出现概率越大，则其包含的信息就越少。

将 X 的各符号 x_i 的信息量 $I(x_i)$ 由概率 $P(x_i)$ 进行加权平均可得 X 的各符号的平均信息量为

$$H(X) = \sum_{i=1}^{n} p(x_i) I(x_i) = -\sum_{i=1}^{n} p(x_i) \log p(x_i) \tag{2—2}$$

在信息论中 $H(X)$ 称为信源 X 的熵。信息论的原理表明，如果知道一个无记忆信源X的熵 $H(X)$，那么总能找到一种信息源的压缩编码方法使表示所有每

个符号所需的平均比特数尽量接近$H(X)$。Shannon信息论认为，信源的平均信息量（熵）就是无失真编码的理论极限。

通常情况下，信源先后发出的消息符号之间是彼此依存的，具有这种特征的信源称为有记忆信源。有记忆信源的极限熵为

$$\lim_{n\to\infty} H_n(x) = \lim_{n\to\infty} H(x_n \mid x_1, x_2 ..., x_{n-1}) \quad (2-3)$$

用H_∞表示。

假定随即变量X_p的取值只与前面的M个符号$X_{p-1}, X_{p-2}, ..., X_{p-M}$有关，称这样的信源为马尔可夫（Markov）信源。并用M阶马尔可夫链表示这M＋1个符号$X_{p-M}, ..., X_{p-2}, X_{p-1}, X_p$所组成的序列。此时可用条件概率分布

$$p(x_i \mid x_{i-1}, x_{i-2}, ..., x_{i-m}, ...) = p(x_i \mid x_{i-1,} x_{i-2}, ..., x_{i-m})$$

描述信源的统计特性。其中，m为阶数，称做记忆阶数。这样算得的平均符号熵可称为H_{M+1}，当m为1时，就是2阶熵值；对于无记忆信源，m为0，得1阶熵值。

无损压缩的目标是除去数据中的冗余，无损压缩技术中基于上下文关系的无损压缩方法在传输和存储文本以及程序文件中有着广泛的应用，以PkZip方法为例，它对文本文件压缩后压缩比可达到6.55，然而这些基于上下文关系的无损压缩方法用于音频压缩，并不能获得理想的效果，PkZip应用于音频信号时压缩比只有1.07。原因在于音频的信源数据不是相互独立的，是一种有记忆的信源，即相邻样点之间存在着很大的冗余度。因此需要探寻适合音频信号的无损压缩方法。

2.2 无损音频压缩的方案

目前，无损音频压缩的方案有两类：可分级编码（含无损编码模块）和独立的无损编码。

可分级编码的系统如图2-1所示。它是基于有损编码算法发展而成，包括有损编码模块和一个无损增强层。码字c是输入信号x经正交变换、量化和熵编码

后的结果。为了达到无损压缩，编码器对c进行本地解码并产生有损重建信号y，将其与输入信号x相减，从而获得差值e，再对e进行无损编码。有损压缩编码c和误差e的无损编码c_e均需传输。在解码器端，有损解码的重建值加上通过对c_e无损解码得到的误差信号e，最终获得对输入信号的完美重建。LTAC[19] (Lossless Transform Audio Codec, 无损变换音频压缩)算法为该方案的代表。上述方案中，可以近似认为有损部分表示原始信号的“轮廓”，而无损部分表示信号的“细节”。因而它特别适合于网络实时应用。当带宽比较宽裕时，可以传输完整的高质量音频；而在带宽受限时便只传输“轮廓”，这样在声音质量和网络带宽之间获得良好的折中。

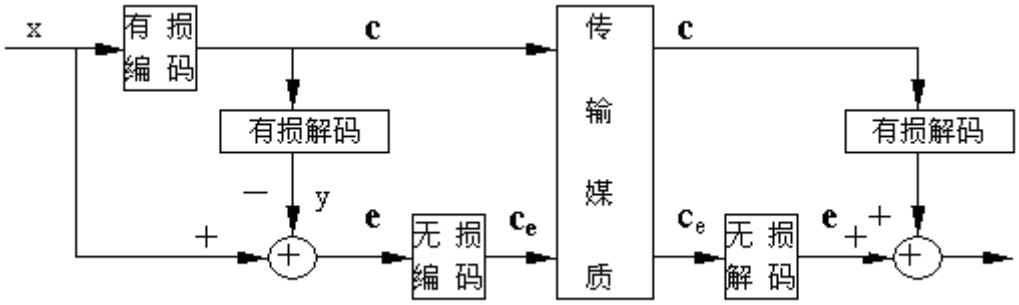


图2－1 可分级的无损编码方案

Figure 2-1 Scalable lossless coding scheme

随着“整数－整数变换”的出现，独立的无损编码方案也随之发展起来，其结构如图2－2所示。音频信号经过分帧后送入变换模块进行整数变换，变换后的系数依然为整数，对这些系数作熵编码，并形成压缩码流。解码器的操作与之相反，先进行熵解码，然后作整数逆变换，恢复出原始数据。由于在整数变换的过程中，使得信号的能量分布更为集中，为熵编码提供了有利条件，因而可以实现较为有效的压缩。

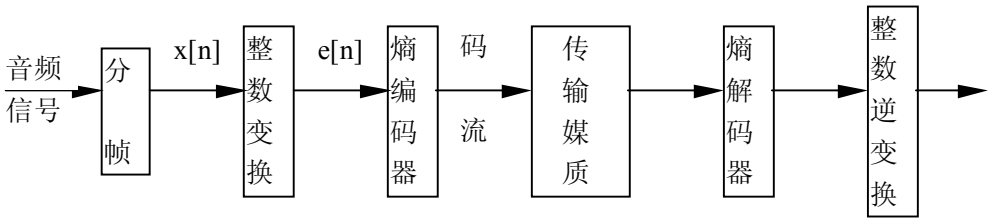


图2-2 独立的无损编码方案

Figure 2-2 Lossless-only coding scheme

两种方案相比，独立的无损编码实现上更为简单。

2.3 无损音频压缩的关键技术

音频信源的冗余度寓于信源间的相关性中。由信息论的原理可知，音频的无损压缩编码可以分为两步：第一步是去相关，用于降低音频数据的动态范围，去除冗余信息，第二步是编码，采用熵编码器对残差数据进行编码。

建立信源模型的目的是建立一个统计模型，能使估计符号概率接近编码信源的概率。如预测模型或变换模型等对音频数据进行预测或变换来去除它们之间的相关性使之成为不相关信源。

目前，熵编码技术已经可接近数据的信息熵，而去相关技术仍有很大的研究余地。无损音频压缩算法已从单纯的熵编码发展为基于一定的模型去相关后再进行熵编码的方法。

图2—3是无损音频压缩算法典型框图。音频压缩的基本方案：首先将信号分帧，在一帧时间内将音频信号作为短时平稳信号进行处理；去除音频信号样点之间的冗余信息；用有效的编码方案进行编码。

图2—3 无损音频压缩算法典型框图

Figure 2-3 Basic operations in most lossless compression algorithms

下面分析无损编码的几个关键技术。

2.3.1 分帧

音频信号特性是随时间而变化的，是一个非平稳的随机过程。但是，从另一方面看，虽然音频信号具有时变的特征，但在一个短时间范围内其特性基本保持不变。将音频看作是一个准平稳过程，对音频的分析和处理都必须建立在“短时”的基础上，即进行“短时分析”。因此无损压缩的第一步是分帧操作，将数字音频信号划分为相同长度的段来分析，每一段称为一帧。在一帧时间内将音频信号作为短时平稳信号进行处理，也便于对压缩后的比特流进行编辑，实现随机访问。随即访问可以快速地访问音频编码后的比特流的任一部分，而

不需要解码之前的部分。帧的持续时间不能太短，否则编码算法的计算复杂度和用于传输头信息的开销会很大。头信息非常重要，因为它根据每帧输入信号的特征定义了该帧编码算法的参数，而且根据不同的应用场合，头信息中还包含了一些附加的数据，如多媒体的同步信息。另外，帧的持续时间也不能太长，否则无法保证信号的短时平稳性，由于长时间内样点变化剧烈导致预测模型或者变换模型失效，从而无法充分去除样点间的相关性。因此，确定适当的帧长是很重要的。由于音频通常在10－30ms之内是保持相对平稳的，因而帧长一般取10－30ms[40]。一些算法的仿真结果表明[24][34][35][36]，一帧持续时间在13到26ms之间，对于44.1kHz的采样率，对应帧长576到1152个样点比较合适。

另外为了更好的适应音频信号的快速变化，可根据实际情况将每帧再划分成若干个短块。在LTAC算法[5]的自适应块长模式中，以每M个样点为一帧，压缩时分别计算变换长度为M，M/2和M/4点各种块的组合方式，从中选择获得最优压缩比的组合方式。

2.3.2 预测编码

预测编码是数据压缩技术的一个重要分支，其理论基础主要是现代统计学和控制论。音频的相邻样点之间存在着高度的相关性，即原始数据中存在着大量的冗余。去除这些冗余，信号就可以有效的压缩编码。预测技术是最简单和有效的压缩方法。在已经被传送（接收）的样点的基础上预测当前样点值，假设以 $x(n)$ 表示当前样点值，以 $\hat{x}(n)$ 表示其预测值，预测残差 $e(n)=\hat{x}(n)-x(n)$ 需要被传送。如果预测相当精确，那么预测残差的分布就集中在零值附近，残差信号的零阶熵比原始信号的零阶熵大为降低。

预测编码有线性预测和非线性预测两类。其中线性预测（LPC）技术广泛用于音频信号处理的各个方面，其基本思想是由于音频样点之间存在的相关性，可以用过去的样点值来预测现在或未来的样点值,即一个音频样点能够用过去若干个音频样点或它们的线性组合来逼近。

当前时域信号采样 $x(n)$ 可以用过去的时域信号采样的预测值来逼近：

估计信号：

$$\hat{x}(n)=\sum_{k=1}^{K}h_k x(n-k) \qquad (2-4)$$

k是预测器的阶数，预测误差信号为：

$$e(n)=\hat{x}(n)-x(n) \qquad (2-5)$$

在前向线性预测器中,求解最优的预测器系数有三种方法：自相关法、协相关法和格形法。自相关法可通过莱文逊—杜宾(Levinson - Durbin)递推算法来高效的求解。

预测编码的关键在于预测算法的选取，这与信号的概率分布有很大关系，实际中常根据大量的统计结果采用简化的概率分布形式来设计最佳的预测器，有时还使用自适应预测器以较好的刻画信号的局部特性，提高预测效率。

2.3.3 变换编码

另一种去相关的方法是正交变换。信息论的研究表明，正交变换不改变信源的熵值，由于去除了大部分相关性，在变换域中的信息熵为高阶信息熵，使用变长编码就能达到高阶熵编码。此外，统计分析表明，经过正交变换后，信号能量向新坐标系中的少数坐标集中，便于量化。正交变换本身并不压缩数据量，只是为在新坐标系中的数据压缩创造了条件。

虽然完整表示音频需要大量的数据，但由于音频数据是高度相关的，有着大量的冗余信息。音频压缩的目标就是尽可能去除这些冗余信息。通过几十年的努力，人们对音频压缩进行了大量的研究，提出了很多压缩方法，例如熵编码，预测编码，变换编码，子带编码等等。变换编码由于其优异的压缩性能，在音频压缩中得到了广泛的应用。

变换编码不直接对时域的音频信号编码，而是首先将时域信号按照某种特定的正交变换或者非正交变换映射到另一个矢量空间，产生一组变换系数，然后对这些系数进行量化和熵编码。一个最优的变换应该是用最少的比特数得到信号最好的变换域表示，经常提及的变换有：karhunen-Loeve 变换（KLT）,离散 Fourier 变换（DFT），离散余弦变换（DCT）和改进型离散余弦变换（MDCT）等。在众多的变换编码中，从压缩效果上看 KLT 是一种最佳变换，其最佳性表

现在：（1）变换后系数互不相关；（2）数值较大的方差只出现在少数系数中，即在变换域中能量是高度集中的，这样就有可能在允许的失真度内将数据量压缩到最小。虽然 KLT 是统计上的最佳变换，但在实现这个变换时仍存在许多困难：首先，KLT 是由信号统计特征唯一确定的，当信号的统计特性不同时，必须重新计算 KLT 矩阵；其次，尽管 KLT 对于简单的统计模型可以用已知的解析式表示，但是一般来说，KLT 计算是没有可行的快速算法。因此，人们转而寻找能够实时处理的次最佳变换。DCT 是一种最接近 KLT 的正交三角函数基变换。由于 DCT 算法的计算复杂度适中，具有多种快速算法，故在图像数据压缩中应用非常广泛。

由于 DCT 等正交变换在边界处存在着固有的不连续性（只不过不同类型的正交变换，其不连续性的程度不同），因此在这些分组边界处就可能出现边界效应。所谓的“块边界效应”是指由于传统的块变换编码对每个块独立处理，对不同块的变换域系数的量化精度不同，引入的量化噪声特性也不同。这样重建出的时域信号在块交界处会导致感觉上的不连续性。这种现象称为块边界效应。为了消除块边界效应，提出了改进型离散余弦变换（MDCT）。

MDCT 是当前音频变换编码中的主流变换手段，如 mp3，AC3，AAC 等标准中均使用了 MDCT。MDCT 优点如下：

a) MDCT 具备完美重建特性，在无限精度条件下可以准确重建信号
b) MDCT 一次处理 M 个数据，产生 M 个频域系数。因此 MDCT 具有临界抽样的特性，不产生冗余数据，不增加系统的数据负荷
c) MDCT 可以有效的降低编码重建信号的块边界效应
d) MDCT 与 DFT 之间存在直接的联系，MDCT 变换的系数可以有效表示谱信息。

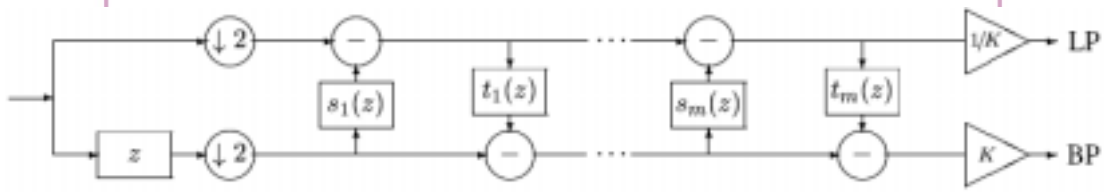
MDCT 是重叠调制变换，可以利用基于 FFT 的快速算法，也可以利用基于 DCT 的快速算法来实现。

以上讨论的变换编码的方法都是浮点运算，然而浮点运算有不可克服的缺点如：运算舍入带来的精度误差导致变换无法实现完全可逆，运算速度慢等。在提升方案提出之前就有人从事整数变换的研究，但由于涉及到多个数的取整问题，很难推导其逆变换，所以一直没有太大的进展。直到Sweldens提出提升

方案，整数变换才真正得到发展。无损音频压缩中常使用的整数变换有IntMDCT（Integer Modified discrete Cosine Transform，整数改进型离散余弦变换）和IWT（Integer Wavelet Transform，整数小波变换）。它们的基础理论都来自于Daubechies和Sweldens的提升方法[6][7]。在第三章中本文将专门介绍整数DCT和整数MDCT的实现，下面简单介绍在第二代小波的理论和提升方案框架下的整数到整数小波变换。

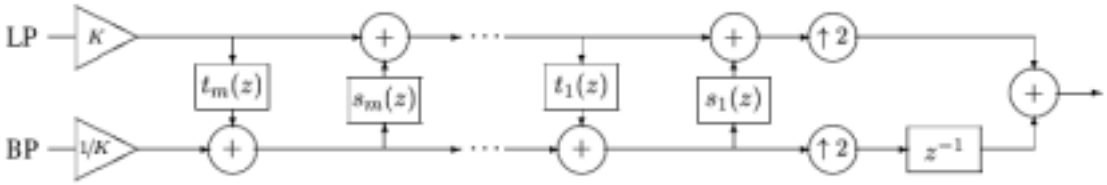
Swenldens等提出的提升算法基于文献[6]的理论，即任何一个具有有限长度滤波器的小波变换均可以通过以下方法得到：把信号分为奇数点和偶数点，然后再经过有限次的提升及尺度缩放可获得可逆的整数小波变换。

文献[7]中讨论了怎样将一个利用已有的小波基计算小波变换的问题转化为利用提升方法来计算小波变换，这样将一个较复杂的小波变换问题转化为一个简单的级联的提升过程，典型结构如图2—4所示。为了方便地写出用提升方法构造整数小波变换的过程，将提升过程概述如下。



(a) 分析过程

(a) Analysis process



(b) 综合过程

(b) Synthesis process

图2—4 传统小波变换的提升实现方案

Figure 2-4 Lifting scheme of tradition wavelet transform

一个传统小波变换分解为提升计算过程，由如下三步组成，具体推导详见文献[7]：

第一步 Lazy小波

$$s_{1,l}^{(0)} = s_{0,2l}$$
$$d_{1,l}^{(0)} = s_{0,2l+1} \quad (2-6)$$

第二步 级联的提升和对偶提升过程

$$d_{1,l}^{(i)} = d_{1,l}^{(i+1)} - \sum_k p_k^{(i)} s_{1,l-k}^{(i-1)}$$
$$s_{1,l}^{(i)} = s_{1,l}^{(i-1)} - \sum_k u_k^{(i)} d_{1,l-k}^{(i)} \quad (2-7)$$

上标i表示第i级提升，$p_k^{(i)}$，$u_k^{(i)}$ 表示提升计算使用的系数，设级联共有M级。

第三步 比例计算

$$s_{1,l} = s_{1,l}^{(M)} / k$$
$$d_{1,l} = d_{1,l}^{(M)} k \quad (2-8)$$

反变换是与上述步骤相反的三步。

注意比例因子k的作用，在进行小波变换时，可以省略正变换的第三步和反变换的第一步，将比例因子的作用融入到量化过程中。

可逆整数小波变换由如下伪码表述：

$s_{1,l}^{(0)} := s_{0,2l}$

$d_{1,l}^{(0)} := s_{0,2l+1}$

for i=1：（1）：M

$$\forall l : d_{1,l}^{(i)} = d_{1,l}^{(i-1)} - \left\lfloor \sum_k p_k^{(i)} s_{1,l-k}^{(i-1)} + \frac{1}{2} \right\rfloor$$

$$\forall l : s_{1,l}^{(i)} = s_{1,l}^{(i-1)} - \left\lfloor \sum_k u_k^{(i)} d_{1,l-k}^{(i)} + \frac{1}{2} \right\rfloor$$

end

反变换由如下伪码描述：

for i=M：（－1）：1

$$\forall l : s_{1,l}^{(i-1)} = s_{1,l}^{(i)} - \left\lfloor \sum_k u_k^{(i)} d_{1,l-k}^{(i)} + \frac{1}{2} \right\rfloor$$

$$\forall l : d_{1,l}^{(i-1)} = d_{1,l}^{(i)} - \left\lfloor \sum_k p_k^{(i)} s_{1,l-k}^{(i-1)} + \frac{1}{2} \right\rfloor$$

end

2.3.4 熵编码

熵编码器用于消除变换后系数的冗余度，它本身并不引入失真。最常用的熵编码方法是Huffman编码。它利用输入符号的先验概率，概率大的符号使用短码表示，概率小的符号使用长码表示，从而实现信息的压缩。精心设计的Huffman码本，其输出的平均码长接近信源的信息熵，即码长的下限。

常用的熵编码方法有游程编码RLC（Run Length Coding）、霍夫曼编码（Huffman Coding）、算术编码以及Golomb-Rice编码等等。在无损编码中，由于音频类型千差万别，设计出的Huffman码本很难适合于各种类型音频的统计特性，无法获得良好的编码增益。因此考虑采用其他的方法，这里主要介绍三种——算术编码、Golomb-Rice码、Elias-Gamma码。

首先定义两种基本码： α (n)和 β (n)。

- α (n)是一元码，用n个0后接1（或者n个1后接0）表示n
- β (n)是二元码，用n的自然二进制表示n，从数值的最高位的1开始

2.3.4.1 算术编码

Huffman码字必定是整数比特长，有时就成为一个问题，如一个符号的概率占90%，根据信息论中信息量 $I(x_i)$ 的定义

$$I(x_i) = -\log_2 p(x_i) = -\log_2 0.9 = 0.15\ bit \qquad (2-9)$$

则该字的最优码长为0.15bit，huffman编码体制可能给这个符号设定一个1bit的码字，它比所需的长度数倍还长。

算术编码是 80 年代发展起来的一种熵编码方法，它避开了用一个特定码字代替一个输入符号的思想，用一个单独的浮点数来代替一串输入符号。其基本

原理是任何一个数据序列均可表示成 0 到 1 之间的一个间隔，该间隔的长度为该序列的出现概率。再选择该间隔内一个代表性的二进制数作为实际的编码输出（通常选择最左端的概率作为二进制数）。可以根据信源的统计特性来设计具体的编码器，也可以针对未知概率的信源设计能够自适应适配其分布的算术编码器。有关数据表明[38]，在未知信源概率分布的大部分情形下，算术编码要优于 Huffman 编码。

算术编码有两个关键[37]，当前符号的概率和间隔。下面举例予以说明。假设有四个符号，00，01，10，11，概率分别为 0.1，0.3，0.4，0.2。然后我们为每个符号分配概率间隔：

符号	概率	间隔
00	0.1	[0, 0.1)
01	0.3	[0.1, 0.4)
10	0.4	[0.4, 0.8)
11	0.2	[0.8, 1.0)

假设我们有一条消息：01，11，10；当每条消息被处理的时候，分配给该符号的间隔比例就变窄，随着编码过程的进行，间隔就越来越窄。处理过程说明如下：

[0, 0.1)　[0.1, 0.4)　[0.4, 0.8)　[0.8, 1.0)

01→间隔 ＝ [0.1, 0.4)

11→间隔 ＝ [0.1＋(0.4-0.1)*0.8, 0.1+(0.4-0.1)*1.0)

＝ [0.1+0.24, 0.1+0.3)

＝[0.34, 0.4)

00→间隔 ＝ [0.34＋(0.4-0.34)*0, 0.34+(0.4-0.34)*0.1)

＝ [0.34, 0.34+0.006)

＝[0.34, 0.346)

因此，消息被编码为间隔[0.34, 0.346)

2.3.4.2 Golomb-Rice 码

近几年发展起来的Golomb-Rice码是一种特殊的Huffman编码方法，是一个

对较宽的熵条件都有效和可以迅速自适应的无损压缩算法，它在编码时不需要码表，但却能提供与使用多个霍夫曼码表相同的功能[13] [17]。经验认为，当编码信号呈Laplacian分布时，Golomb-Rice码算法性能优于Huffman编码和算术编码，因而该编码方法在无损压缩领域引起了广泛的关注。

Golomb码是一种无符号整型最佳指数分布编码。对于一个整数n，用其商和余数来描述，除数m为其参数。如果m是2的幂次，n的码字是前缀α(n/m)和用log m比特二进制表示的n mod m的简单串联而成的，即α(n/m): β(n mod m)。若m不是2的幂次，则可令k为满足2k≥2m的最小正整数。码表包括码字长度≥k的m个码字和长度为k－1的2k-1-m个码字；令j=2k-1-m，j个码字用k－1比特表示，剩下的m个码字用k比特表示。所以，整数n的Golomb（m）码由两个码 $\alpha((n-j)/m):\beta((n-j)\%m+2j)$ 串联得到。例如：m＝4，，n＝13，商和余数分别为3和1，即α（3）：β（1），13的编码111001。

Rice码是一种简化的Golomb码，其参数为k且m=2k（Rice（k）码即为Golomb（2k）码）。整数n的Rice码表示为α（n div m）与β（n mod m）的串联，总共 $\frac{n}{2^k}+k+1$ 个比特。

Rice编码简单快速，而算术编码有较高的计算复杂度；另一方面，Rice编码器在编码效率上也较接近算术编码；再者，使用Rice编码器不涉及专利问题。对于按拉普拉斯分布的信源，Rice算法性能优于Huffman编码和算术编码[41]。

2.3.4.3 Elias-Gamma 码

Elias γ码是组成β码（二进制表示）的比特逆序排列而成，用标志位“0”来分割每一个比特。标志位“1”来表示数值的最高比特位。例如，13用二进制表示为1101，逆序后为1011，每一比特之间加上标志位，其Elias γ码为0100011，其中下划线标识的为标志位，最后的比特“1”既是标志位又是数据位。γ’码是γ码的一种变型，标志位（α码）后接数据位（β码）。13的γ’码为0001101（α码结束位的“1”又是β码的第一个比特）。大多数的文献中“Elias γ码”通常是指γ’码。

表 2-1　不同编码方法的码字长度的比较

Table 2-1 Comparison of various representations - codeword lengths

值	二进制	Elias γ	Golomb			Rice		
			m=2	m=3	m=4	k=2	k=3	k=4
1	1	1	2	3	3	3	4	5
2	2	3	2	3	3	3	4	5
4	3	5	4	4	4	4	4	5
10	4	7	7	6	5	5	5	5
20	5	9	12	9	8	8	6	6
50	6	11	27	12	15	15	10	8
200	8	15	102	69	53	53	29	17
500	9	17	252	169	128	128	66	36
1000	10	19			253	253	129	67
2000	11	21				503	254	130
5000	13	25					629	317
10000	14	27						630

表2－1给出上述几种编码的码字长度[8]。根据其中的数据，Golomb-Rice码对于中等取值的数比较有效，但对数值偏大和偏小的数字没有其他方法好。

2.4 本章小结

本章从信息论的角度介绍了无损音频压缩的原理，指出当前无损音频压缩两种不同的技术方案：可分级的无损编码方案和独立的无损编码方案。接着阐述当前无损音频算法中所采用的一些关键技术——分帧、预测编码、变换编码和熵编码，并深入分析这些技术各自的优缺点，为提出新的无损音频压缩算法提供理论依据和指导。

第三章 MDCT 变换的整数实现算法

离散余弦变换（DCT，discrete cosine transform）由于其良好的去除数据相关能力和低计算复杂度的特点，已被采纳为图像压缩标准和视频压缩标准的主要变换。但是 DCT 用于音频压缩时，由于存在着严重的块边界效应，产生很大的噪声，限制了其应用，因此提出改进型离散余弦变换（MDCT）用于消除块边界效应。

然而传统的MDCT基于浮点算法设计，在定点实现时由于存在截断误差，往往无法实现完全可逆。因此期望研究整数MDCT，用整数变换矩阵代替原来的浮点数变换矩阵，或者采用完全可逆的运算，将整数序列映射为整数变换系数，使之实现真正的可逆。

如前所述，整数变换的基础理论来自于Daubechies和Sweldens的提升方法[6][7]。由于DCT和MDCT之间有密切的联系，而提升技术是实现整数DCT的重要方法，因此下面首先介绍两种提升方法，然后再分析整数DCT的实现，在此基础上实现整数MDCT。

3.1 提升格式

3.1.1 经典提升格式

首先，以上三角矩阵为例来简述提升的思想。若变换:

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} 1 & \alpha \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \quad (3-1)$$

其中 x_1, x_2 为输入, y_1, y_2 为输出, α 为浮点数， 则可构造出其对应的整型变换

$$\begin{aligned} y_1 &= x_1 + [\alpha x_2] \\ y_2 &= x_2 \end{aligned} \quad (3-2)$$

其中$[x]$表示对 x 取整。从式(3－2) 可以看出，如果 x_1, x_2 为整数，那么经

过计算得到的y_1, y_2也必定为整数。因此，式(3－2) 确实是一个从整数到整数的变换. 同时，可以得到变换(3－2) 的逆变换：

$$\begin{aligned} x_2 &= y_2 \\ x_1 &= y_1 - [\alpha x_2] \end{aligned} \qquad (3-3)$$

式(3－3)表示两个整数y_1, y_2可以先通过y_2 重建x_2，然后再由y_1和x_2重建x_1。 易知，式(3－3) 也是整型的变换，且该变换可逆。与之类似，下三角型矩阵也能够找到其对应的可逆变换。

如果变换矩阵可以分解成多个主对角线元素为1或－1 的三角矩阵的乘积, 那么，只要对每一个三角矩阵分别找到其对应的可逆整型变换，然后按分解顺序依次变换，就可构造出整个变换的整型变换。文献[7]就是利用这种思想寻找小波变换所对应的可逆整型变换。这类方法统称为提升，而主对角线元素为1或者－1的三角矩阵称为提升矩阵。

可逆整数变换保证变换对于整数信号是真正可逆的，同时其结果与浮点运算结果相差无几，能够有效地降低信号的冗余度，从而为数据的压缩提供有利条件。

音频压缩中经常使用的改进型离散余弦变换（MDCT）可以完全分解为Givens旋转，下面给出Givens旋转分解为提升的基本公式：

$$\begin{pmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{pmatrix} = \begin{pmatrix} 1 & \frac{\cos(\alpha)-1}{\sin(\alpha)} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ \sin(\alpha) & 1 \end{pmatrix} \begin{pmatrix} 1 & \frac{\cos(\alpha)-1}{\sin(\alpha)} \\ 0 & 1 \end{pmatrix} \qquad (3-4)$$

Givens旋转的整数近似：

$$\begin{aligned} y_1^{(0)} &= x_1 + [-tg(\alpha/2)x_2] \\ y_2^{(0)} &= x_2 \\ y_1^{(1)} &= y_1^{(0)} \\ y_2^{(1)} &= y_2^{(0)} + [\sin(\alpha)y_0^{(0)}] \\ y_1 &= y_1^{(1)} + [-tg(\alpha/2)y_2^{(1)}] \\ y_2 &= y_2^{(1)} \end{aligned}$$

则其对应的逆变换为

$$y_2^{(1)} = y_2$$
$$y_1^{(1)} = y_1 - [-tg(\alpha/2)y_2^{(1)}]$$
$$y_1^{(0)} = y_1^{(1)}$$
$$y_2^{(0)} = y_2^{(1)} - [\sin(\alpha)y_0^{(0)}]$$
$$x_2 = y_2^{(0)}$$
$$x_1 = y_1^{(0)} - [-tg(\alpha/2)x_2]$$

上述过程均为整数运算，由正逆变换可以看出，该变换可以完全可逆。易知，一个Givens旋转需要3次取整操作。

3.1.2 多维提升格式

拉伸（scaling）矩阵的提升格式为：

$$\begin{pmatrix} d & 0 \\ 0 & d^{-1} \end{pmatrix} = \begin{pmatrix} -1 & 0 \\ d^{-1} & 1 \end{pmatrix} \begin{pmatrix} 1 & -d \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & d^{-1} \end{pmatrix} \quad (3-5)$$

当式（3－5）中数d用任意一个N×N的矩阵**T**代替，0用N×N的零矩阵**0**代替，1用N×N的单位矩阵$\mathbf{I}_N$代替，则对于下列2N×2N块矩阵，下式是成立的：

$$\begin{pmatrix} \mathbf{T} & \mathbf{0} \\ \mathbf{0} & \mathbf{T}^{-1} \end{pmatrix} = \begin{pmatrix} -\mathbf{I}_N & \mathbf{0} \\ \mathbf{T}^{-1} & \mathbf{I}_N \end{pmatrix} \begin{pmatrix} \mathbf{I}_N & -\mathbf{T} \\ \mathbf{0} & \mathbf{I}_N \end{pmatrix} \begin{pmatrix} \mathbf{0} & \mathbf{I}_N \\ \mathbf{I}_N & \mathbf{T}^{-1} \end{pmatrix} \quad (3-6)$$

对于具有结构$\begin{pmatrix} \mathbf{T} & \mathbf{0} \\ \mathbf{0} & \mathbf{T}^{-1} \end{pmatrix}$的2N×2N块矩阵使用上式的提升方法，称为“多维提升”。

设$\vec{x}_1, \vec{x}_2$表示两个列向量信号，$\vec{y}_1, \vec{y}_2$表示$\vec{x}_1, \vec{x}_2$经整数变换**T**后的输出。设$\vec{x}_1, \vec{x}_2, \vec{y}_1, \vec{y}_2$，$\vec{r}_1^{(0)}$，$\vec{r}_2^{(0)}$，$\vec{r}_1^{(1)}$，$\vec{r}_2^{(1)}$为N×1的列向量。[g]表示对列向量的每一个元素分别取整。

则$\begin{pmatrix} \vec{r}_{y_1} \\ \vec{r}_{y_2} \end{pmatrix} = \begin{pmatrix} \mathbf{T} & \mathbf{0} \\ \mathbf{0} & \mathbf{T}^{-1} \end{pmatrix} \begin{pmatrix} \vec{r}_{x_1} \\ \vec{r}_{x_2} \end{pmatrix}$的整数变换如下：

$$\begin{aligned}
\vec{y}_1^{(0)} &= \vec{x}_2 \\
\vec{y}_2^{(0)} &= \vec{x}_1 + [\mathbf{T}^{-1} g \vec{x}_2] \\
\vec{y}_1^{(1)} &= \vec{y}_1^{(0)} - [\mathbf{T} g \vec{y}_2^{(0)}] \\
\vec{y}_2^{(1)} &= \vec{y}_2^{(0)} \\
\vec{y}_1 &= -\vec{y}_1^{(1)} \\
\vec{y}_2 &= \vec{y}_2^{(1)} + [\mathbf{T}^{-1} g \vec{y}_1^{(1)}]
\end{aligned}$$

逆变换为：

$$\begin{aligned}
\vec{y}_2^{(1)} &= \vec{y}_2 - [\mathbf{T}^{-1} g \vec{y}_1^{(1)}] \\
\vec{y}_1^{(1)} &= -\vec{y}_1 \\
\vec{y}_2^{(0)} &= \vec{y}_2^{(1)} \\
\vec{y}_1^{(0)} &= \vec{y}_1^{(1)} + [\mathbf{T} g \vec{y}_2^{(0)}] \\
\vec{x}_2 &= \vec{y}_1^{(0)} \\
\vec{x}_1 &= \vec{y}_2^{(0)} - [\mathbf{T}^{-1} g \vec{x}_2]
\end{aligned}$$

这样，整个变换过程都是整数运算，只要正反变换中采用相同的取整方式，变换过程完全可逆。

3.2 整数 DCT

3.2.1 经典提升实现整数 DCT

由上述所述的 Givens 旋转的提升方法，可以考虑将 DCT 完全分解为 Givens 旋转，由 Givens 旋转的提升格式来实现整数 DCT。下面介绍如何将 DCT 完全分解为 Givens 旋转。

IV 型 DCT 定义如下[10]：

$$X(m) = \sqrt{\frac{2}{N}} \sum_{n=0}^{N-1} x(n) \cos \frac{(2n+1)(2m+1)\pi}{4N}, \quad m = 0,1,2,\ldots,N-1$$

由于涉及到变换的分解，本论文用矩阵的表示方式来说明会比较清楚。黑体的符号表示一个矩阵，其阶数用下标表示，如果矩阵是方阵，下标只标出行（列）向量的维数。N 是 2 的整数次幂。下面定义 IV 型 DCT 的变换核矩阵为：

$$\mathbf{C}_N^{IV} = \sqrt{\frac{2}{N}}\left[\cos((m+\frac{1}{2})(n+\frac{1}{2})\pi / N)\right] \quad m,n = 0,1,...,N-1 \qquad (3-7)$$

并且 IV 型 DCT 变换核矩阵的逆也是该矩阵

即 $\mathbf{C}_N^{IV-1} = \mathbf{C}_N^{IV}$ （3－8）

Chen et al.首先在文献[8]中给出了 $\mathbf{C}_N^{IV}$ 的分解，由 WangZhongde 提出修正[9]。然而文献[9]中的下标定义过于繁琐，下面的下标表示更加简洁。

令 $J=\log_2 N$

$$\mathbf{C}_N^{IV} = \mathbf{Q}_N \mathbf{V}_N(J)\mathbf{U}_N(J-1)\mathbf{V}_N(J-1)\cdots\mathbf{U}_N(1)\mathbf{V}_N(1)\mathbf{H}_N \qquad (3-9)$$

这些矩阵有 5 种不同的类型。

类型 1：第一个矩阵是置换矩阵，其作用是将下标为奇数的分量逆序

$$\mathbf{Q}_N = \begin{pmatrix} 1 & 0 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 0 \\ 0 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 0 & 1 \\ 0 & 0 & 1 & 0 & \cdot & \cdot & \cdot & \cdot & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & 1 & 0 & \cdot & \cdot & \cdot & 0 \\ 0 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & 0 \\ 0 & 1 & 0 & \cdot & \cdot & \cdot & \cdot & \cdot & 0 \end{pmatrix} \qquad (3-10)$$

类型 2：最后一个矩阵 $\mathbf{H}_N$ 同样是一个置换矩阵

$\mathbf{H}_N$ 可以表示为

$$\mathbf{H}_N = \mathbf{P}_N \begin{pmatrix} \mathbf{P}_{N/2} & \\ & \overline{\mathbf{P}}_{N/2} \end{pmatrix} \begin{pmatrix} \mathbf{P}_{N/4} & & & \\ & \overline{\mathbf{P}}_{N/4} & & \\ & & \mathbf{P}_{N/4} & \\ & & & \overline{\mathbf{P}}_{N/4} \end{pmatrix} \cdots \begin{pmatrix} \mathbf{P}_4 & & & & & \\ & \overline{\mathbf{P}}_4 & & & & \\ & & \cdot & & & \\ & & & \cdot & & \\ & & & & \mathbf{P}_4 & \\ & & & & & \overline{\mathbf{P}}_4 \end{pmatrix} \qquad (3-11)$$

对于 N≥4

其中

$$\mathbf{P}_j = \begin{pmatrix} 1 & 0 & . & . & . & . & . & . & . & . \\ 0 & . & . & . & . & . & . & . & . & 1 \\ 0 & 1 & . & 0 & . & . & . & . & . & . \\ 0 & . & . & . & . & . & . & . & 1 & 0 \\ . & . & . & . & . & . & . & . & . & . \\ 0 & . & . & . & . & 1 & 0 & . & . & . \\ 0 & . & . & 1 & 0 & . & . & . & . & . \\ 0 & . & . & . & 1 & 0 & . & . & . & . \end{pmatrix} \qquad (3-12)$$

其中 矩阵上两横运算表示将矩阵的行和列全部逆序排列

即矩阵 A 的两横运算表示为

$$\overline{\overline{\mathbf{A}}} = \bar{\mathbf{I}} \cdot \mathbf{A} \cdot \bar{\mathbf{I}} \qquad (3-13)$$

$\bar{\mathbf{I}}$ 是反单位矩阵

$$\bar{\mathbf{I}} = \begin{pmatrix} & & & 1 \\ & & N & \\ & N & & \\ 1 & & & \end{pmatrix} \qquad (3-14)$$

易见，从 $\mathbf{H}_N$ 的表示式求置换矩阵特别是阶数较大时非常繁琐。本文考虑寻求计算 $\mathbf{H}_N$ 的简便方法。

由 $\mathbf{H}_N$ 的表示式，可以得到 $\mathbf{H}_4$

$$\mathbf{H}_4 = \mathbf{P}_4 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

经过 $\mathbf{H}_4$ 置换向量 $\vec{w} = (w_0 \quad w_1 \quad w_2 \quad w_3)^T$ 后

$$\mathbf{H}_4 \cdot \vec{w} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \end{pmatrix} = \begin{pmatrix} w_0 \\ w_3 \\ w_1 \\ w_2 \end{pmatrix}$$

又 4 阶的哈达玛变换矩阵以及行向量变号次数为

变号次数

$$\mathbf{Hada}_4=\frac{1}{2}\begin{pmatrix}1&1&1&1\\1&-1&1&-1\\1&1&-1&-1\\1&-1&-1&1\end{pmatrix}\begin{matrix}0\\3\\1\\2\end{matrix}$$

因此经过归纳推理可以得到：$\mathbf{H}_N$ 置换后的顺序即为哈达玛顺序（哈达玛变换核矩阵的行向量的变号次数顺序）。由于哈达玛变换核矩阵存在递推关系

$$\mathbf{Hada}_2=\frac{1}{\sqrt{2}}\begin{pmatrix}1&1\\1&-1\end{pmatrix}$$

$$\mathbf{Hada}_{2^l}=\frac{1}{\sqrt{2}}\begin{pmatrix}\mathbf{H}_{2^{l-1}}&\mathbf{H}_{2^{l-1}}\\\mathbf{H}_{2^{l-1}}&-\mathbf{H}_{2^{l-1}}\end{pmatrix}=\mathbf{H}_2\otimes\mathbf{H}_{2^{l-1}} \qquad (3-15)$$

$\mathbf{H}_2\otimes\mathbf{H}_{2^{l-1}}$ 表示求矩阵 $\mathbf{H}_2$ 与 $\mathbf{H}_{2^{l-1}}$ 的 Kronecker 积，或者称为矩阵 $\mathbf{H}_2$ 与 $\mathbf{H}_{2^{l-1}}$ 的直积。直积的定义如下：

设 $\mathbf{A}=(a_{ij})_{m\times n},\mathbf{B}=(b_{ij})_{p\times q}$，则 $mp\times nq$ 矩阵

$$\begin{pmatrix}a_{11}B&a_{12}B&\dots&a_{1n}B\\a_{21}B&a_{22}B&\dots&a_{2n}B\\\dots&\dots&\dots&\dots\\a_{m1}B&a_{m2}B&\dots&a_{mn}B\end{pmatrix}$$

称为矩阵 A 和 B 的 Kronecker 积，或者称为 A 与 B 的直积，记为 $\mathbf{A}\otimes\mathbf{B}$，即

$$\mathbf{A}\otimes\mathbf{B}=\begin{pmatrix}a_{11}B&a_{12}B&\dots&a_{1n}B\\a_{21}B&a_{22}B&\dots&a_{2n}B\\\dots&\dots&\dots&\dots\\a_{m1}B&a_{m2}B&\dots&a_{mn}B\end{pmatrix}$$

设 $\mathbf{A}\otimes\mathbf{B}$ 的 (i,j) 位置的元素为 $(a\otimes b)_{ij}$，得

$$(a\otimes b)_{(i_1-1)p+i_2,(j_1-1)q+j_2}=a_{i_1,j_1}b_{i_2,j_2}$$

由上述递推关系（3－15）编程时很容易得到哈达玛变换矩阵，计算出其行向量的变号次数顺序，即经第二个置换矩阵置换的结果，并不需要得到置换矩阵。

类型 3：矩阵 $\mathbf{U}_N(j)$, j=1,2,…,J-1 是块对角二进制矩阵

$$\mathbf{U}_N(j)=\frac{1}{\sqrt{2}}block\ diag\left\{\mathbf{B}_{2^{j+1}}(j),\mathbf{B}_{2^{j+1}}(j),...,\mathbf{B}_{2^{j+1}}(j)\right\} \quad (3-16)$$

其中

$$\mathbf{B}_{2^{j+1}}(j)=\begin{bmatrix}\mathbf{I}_{2^j} & \mathbf{I}_{2^j}\\ \mathbf{I}_{2^j} & -\mathbf{I}_{2^j}\end{bmatrix} \quad (3-17)$$

类型 4：第二个矩阵 $\mathbf{V}_N(J)$ 是一个对角阵

$$[\mathbf{V}_N(J)]=block\ diag\left\{\mathbf{T}(\frac{1}{4N}),\mathbf{T}(\frac{5}{4N}),...,\mathbf{T}(\frac{2N-3}{4N})\right\} \quad (3-18)$$

其中

$$\mathbf{T}_2(r)=\begin{bmatrix}\cos r\pi & \sin r\pi\\ \sin r\pi & -\cos r\pi\end{bmatrix} \quad (3-19)$$

类型 5：所有余下的矩阵 $\mathbf{V}_N(j)$， j=1,2…,J-1,都是块对角阵。它由子矩阵 $\mathbf{I}_{2^j}$ 和 $\mathbf{E}_{2^j}(j)$ 交替出现在自左上到右下的主对角线上

$$\mathbf{V}_N(j)=block\ diag\left\{\mathbf{I}_{2^j},\mathbf{E}_{2^j}(j),\mathbf{I}_{2^j},...,\mathbf{E}_{2^j}(j)\right\} \quad (3-20)$$

其中 $\mathbf{I}_{2^j}$ 表示 $2^j\times 2^j$ 的单位阵，$\mathbf{E}_{2^j}(j)$ 定义为

$$\mathbf{E}_{2^j}(j)=block\ diag\left\{\mathbf{T}_2(\frac{1}{2^{j+2}}),\mathbf{T}_2(\frac{5}{2^{j+2}}),...,\mathbf{T}_2(\frac{2^{j+1}-3}{2^{j+2}})\right\} \quad (3-21)$$

其中 $\mathbf{T}_2(r)$ 的定义如式(3-19)。

注意到所有 2J－1 个矩阵都是对称正交矩阵。由于 $\mathbf{C}_N^{IV}$ 是对称阵，式（3－9）中连乘的顺序可以逆向

$$\mathbf{C}_N^{IV}=\mathbf{H}_N\mathrm{g}\mathbf{V}_N(1)\mathrm{g}\mathbf{U}_N(1)\mathrm{g}\cdot\cdot\mathrm{g}\mathbf{V}_N(J-1)\mathrm{g}\mathbf{U}_N(J-1)\mathrm{g}\mathbf{V}_N(J)\mathrm{g}\mathbf{Q}_N$$

式（3-22）以 $\mathbf{C}_8^{IV}$ 的分解作为例子，其中 $C_r=\cos r\pi, S_r=\sin r\pi$。图 3－1 以 32 点 DCT 为例给出 DCT 完全分解为 Givens 旋转的信号流图。

$$\mathbf{C}_8^{IV}=\sqrt{\frac{2}{8}}\begin{pmatrix}1&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&1\\0&0&1&0&0&0&0&0\\0&0&0&0&0&1&0&0\\0&0&0&0&1&0&0&0\\0&0&0&1&0&0&0&0\\0&0&0&0&0&0&1&0\\0&1&0&0&0&0&0&0\end{pmatrix}\begin{pmatrix}C_{1/32}&S_{1/32}&&&&&&&&&&&&&&\\S_{1/32}&-C_{1/32}&&&&&&&&&&&&&&\\&&C_{5/32}&S_{5/32}&&&&&&&&&&&&\\&&S_{5/32}&-C_{5/32}&&&&&&&&&&&&\\&&&&C_{9/32}&S_{9/32}&&&&&&&&&&\\&&&&S_{9/32}&-C_{9/32}&&&&&&&&&&\\&&&&&&C_{13/32}&S_{13/32}&&&&&&&&\\&&&&&&S_{13/32}&-C_{13/32}&&&&&&&&\end{pmatrix}$$

$$\begin{pmatrix}\mathbf{I}_4&\mathbf{I}_4\\\mathbf{I}_4&\mathbf{I}_4\end{pmatrix}\begin{pmatrix}\mathbf{I}_4&&&\\&C_{1/8}&S_{1/8}&\\&S_{1/8}&-C_{1/8}&\\&&&C_{5/8}&S_{5/8}\\&&&S_{5/8}&-C_{5/8}\end{pmatrix}\begin{pmatrix}\mathbf{I}_2&\mathbf{I}_2&&\\\mathbf{I}_2&\mathbf{-I}_2&&\\&&\mathbf{I}_2&\mathbf{I}_2\\&&\mathbf{I}_2&\mathbf{-I}_2\end{pmatrix}$$

$$\begin{pmatrix}\mathbf{I}_2&&&\\&C_{1/4}&S_{1/4}&\\&S_{1/4}&-C_{1/4}&\\&&&\mathbf{I}_2\end{pmatrix}\begin{pmatrix}\mathbf{I}_2&&&\\&C_{1/4}&S_{1/4}&\\&S_{1/4}&-C_{1/4}&\\&&&\mathbf{I}_2\end{pmatrix}\begin{pmatrix}1&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&1\\0&0&0&1&0&0&0&0\\0&0&0&0&1&0&0&0\\0&1&0&0&0&0&0&0\\0&0&0&0&0&0&1&0\\0&0&1&0&0&0&0&0\\0&0&0&0&0&1&0&0\end{pmatrix}$$

（3－22）

3.2.2 多维提升实现整数 DCT

基于变换的无损音频压缩的整个技术框架是围绕整数变换展开的。因此整数变换的性能非常重要，主要从 2 个方面来评价其性能：运算复杂度和对变换系数的逼近程度。对变换系数的逼近程度主要受取整次数的影响，显然减少取整次数能增加逼近程度，同时也能相应地减少复杂度。

通常地，将四型离散余弦变换（DCT_{IV}）的变换矩阵分解为多个 Givens 旋转，对每一次旋转作整数近似得到整数 DCT_{IV}。N 点 DCT_{IV} 的变换矩阵一般分解为$\left(\frac{N(N-1)}{2}\right)$个 Givens 旋转，而每个 Givens 旋转需要 3 次取整，即需要$\left(\frac{3N(N-1)}{2}\right)$次取整，因此取整次数较多，与浮点结果的误差较大，无法准确的反映信号的频谱特征。

因此考虑使用“多维提升[11]”的思想实现整数DCT。

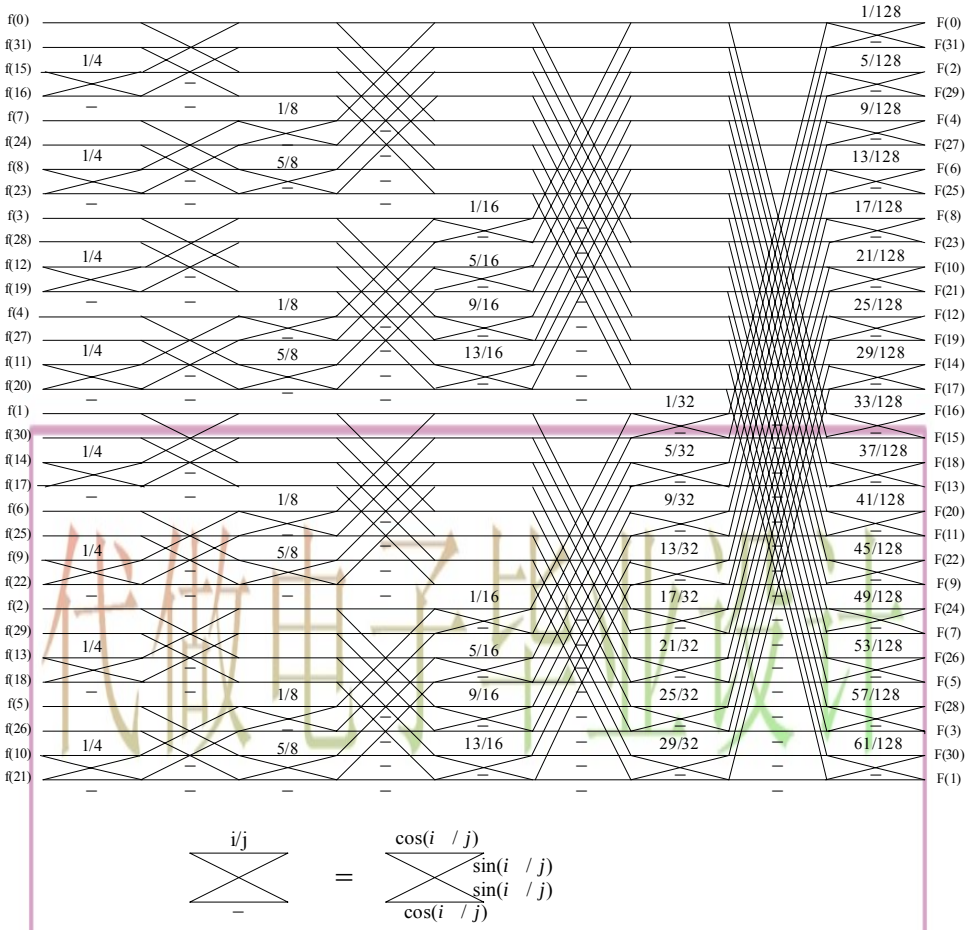


图 3－1　$\mathbf{C}_{32}^{IV}$ 的信号流图

Figure 3-1 Signal flow chart of $\mathbf{C}_{32}^{IV}$

3.2.2.1 立体声

该方法同时对两块信号作整数 DCT_{IV} 变换，这两块信号可以是来自前后两帧或者是立体声的左右声道。设用 **D** 表示 DCT_{IV} 的变换矩阵，由于 **D** 的逆矩阵仍是 **D**，即 $\mathbf{D}_{N\times N}g\mathbf{D}_{N\times N}=\mathbf{I}_N$。$\mathbf{I}_N$ 表示 N×N 的单位阵。基于多维提升的思想，式（3－6）有如下分解形式：

$$\begin{pmatrix}\mathbf{D} & \mathbf{0}\\ \mathbf{0} & \mathbf{D}\end{pmatrix}=\begin{pmatrix}-\mathbf{I}_N & \mathbf{0}\\ \mathbf{D} & \mathbf{I}_N\end{pmatrix}\begin{pmatrix}\mathbf{I}_N & -\mathbf{D}\\ \mathbf{0} & \mathbf{I}_N\end{pmatrix}\begin{pmatrix}\mathbf{0} & \mathbf{I}_N\\ \mathbf{I}_N & \mathbf{D}\end{pmatrix} \tag{3－23}$$

设 $\vec{x}_1,\vec{x}_2$ 表示两块信号，$\vec{y}_1,\vec{y}_2$ 表示 $\vec{x}_1,\vec{x}_2$ 整数DCT_{IV}变换后的输出。

$\vec{x}_1,\vec{x}_2,\vec{y}_1,\vec{y}_2$，$\vec{y}_1^{(0)}$，$\vec{y}_2^{(0)}$，$\vec{y}_1^{(1)}$，$\vec{y}_2^{(1)}$ 为N×1的列向量。[g]表示对列向量的每一个元素分别取整。

则$\begin{pmatrix} \mathbf{r} \\ y_1 \\ \mathbf{r} \\ y_2 \end{pmatrix}=\begin{pmatrix} \mathbf{D} & \mathbf{0} \\ \mathbf{0} & \mathbf{D} \end{pmatrix}\begin{pmatrix} \mathbf{r} \\ x_1 \\ \mathbf{r} \\ x_2 \end{pmatrix}$的整数变换如下：

$$\begin{aligned}
&\mathbf{r}_1^{(0)} = \mathbf{r}_2 \\
&\mathbf{r}_2^{(0)} = \mathbf{r}_1 + [\mathbf{D}\mathbf{g}\mathbf{r}_2] \\
&\mathbf{r}_1^{(1)} = \mathbf{r}_1^{(0)} - [\mathbf{D}\mathbf{g}\mathbf{r}_2^{(0)}] \\
&\mathbf{r}_2^{(1)} = \mathbf{r}_2^{(0)} \\
&\mathbf{r}_1 = -\mathbf{r}_1^{(1)} \\
&\mathbf{r}_2 = \mathbf{r}_2^{(1)} + [\mathbf{D}\mathbf{g}\mathbf{r}_1^{(1)}]
\end{aligned}$$

可以看到：上述过程中均为整数运算，且完全可逆。图 3-2 说明了两块 4 点的整数 DCT_{IV} 实现的具体过程。

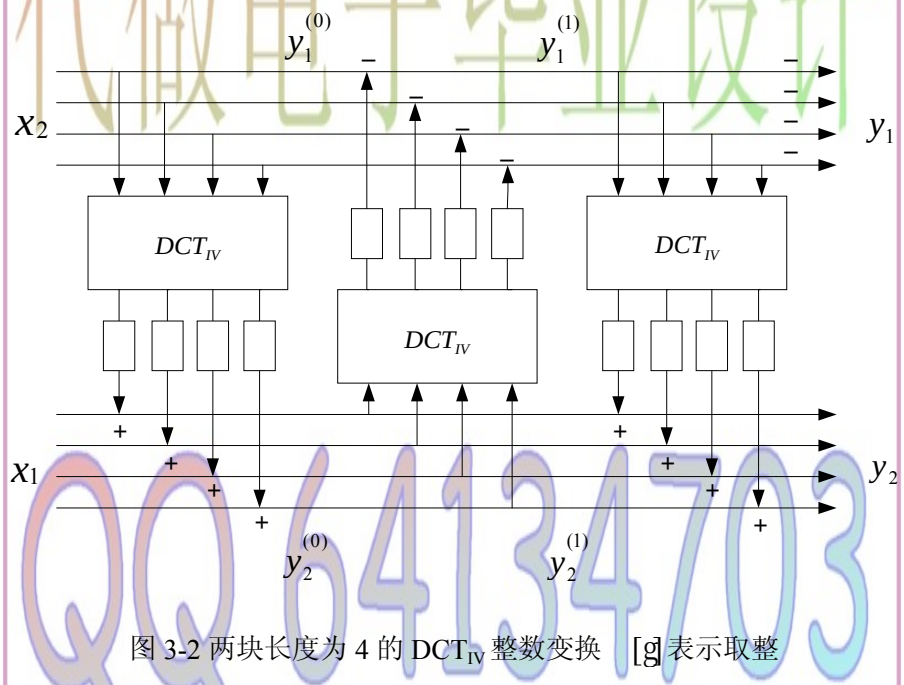


图 3-2 两块长度为 4 的 DCT_{IV} 整数变换 [g]表示取整

Figure 3-2 Invertible integer approximation of two blocks of 4-pointsDCT_{IV}

[g] means rounding operations

由图 3—2 可知，两块长度为 N 的 DCT_{IV} 整数变换，仅需 3N 次取整步骤，即每一变换中仅使用 3N/2 的取整步骤（相对于分解为$\left(\frac{N(N-1)}{2}\right)$个 Givens 旋转）。取整次数减少，与浮点结果的误差也大大减少，更准确地反映了信号的频谱特征，为信号的压缩提供有利的条件。该方法可以用于变换阶数较高的场合，且与浮点结果的误差很小，唯一的不足是将引入信号处理的时延。

3.2.2.2 单声道

立体声 DCT 方法要求同时计算两块的 DCT_{IV} 变换，如计算相邻块的 DCT_{IV} 或者同时计算左右声道的 DCT_{IV}。然而第一种方法给系统引入了一个块的额外时延，第二种方法只能用于立体声信号。如果要求既无时延又非立体声处理，多维提升的思想依然是可行的，但是需要附加一些 Givens 旋转。

长度 N 的 DCT_{IV} 变换核矩阵：

$$\mathbf{DCT}_N^{IV}=\left(\sqrt{\frac{2}{N}}\cos\frac{(2k+1)(2l+1)\pi}{4N}\right)\quad k,l=0,\ldots,N-1$$

可以分解为两个长度为 N/2 的 DCT_{IV} 和前后置换步骤：

$$\mathbf{DCT}_N^{IV}=\mathbf{L}_N\begin{pmatrix}\mathbf{DCT}_{N/2}^{IV} & \mathbf{0}_{N/2}\\ \mathbf{0}_{N/2} & \mathbf{DCT}_{N/2}^{IV}\end{pmatrix}\mathbf{M}_N\mathbf{Q}_N\mathbf{P}_N$$

其中 N×N 的矩阵 L 和 M 定义为

$$\begin{pmatrix}\mathbf{L}_{k,k} & \mathbf{L}_{k,N-1-k}\\ \mathbf{L}_{N-1-k,k} & \mathbf{L}_{N-1-k,N-1-k}\end{pmatrix}=\begin{pmatrix}\cos\left(\frac{2k+1}{4N}\pi\right) & -\sin\left(\frac{2k+1}{4N}\pi\right)\\ -\sin\left(\frac{2k+1}{4N}\pi\right) & \cos\left(\frac{2k+1}{4N}\pi\right)\end{pmatrix}\quad k=0,\ldots,N/2-1$$

$$\mathbf{L}_{k,l}=0\qquad else \tag{3-24}$$

和

$$\mathbf{M}=\frac{1}{\sqrt{2}}\begin{pmatrix}\mathbf{I}_{N/2} & \mathbf{I}_{N/2}\\ -\mathbf{I}_{N/2} & \mathbf{I}_{N/2}\end{pmatrix} \tag{3-25}$$

N×N 置换矩阵 **P** 定义如下：

$$\begin{aligned}&\mathbf{P}_{4k,4k}=\mathbf{P}_{4k+1,4k+1}=\mathbf{P}_{4k+2,4k+3}=\mathbf{P}_{4k+3,4k+2}=1 && k=0,\ldots,N/4-1\\ &\mathbf{P}_{k,l}=0 && else\end{aligned} \tag{3-26}$$

经过置换矩阵 **P** 将每第二对分量的值交换，以 $\underset{\sim}{w}=(w_0\ w_1\ w_2\ w_3\ w_4\ w_5\ w_6\ w_7)^T$ 为例，经过置换矩阵 **P**，得到

$$\mathbf{P}\vec{w}=\begin{pmatrix}1&0&0&0&0&0&0&0\\0&1&0&0&0&0&0&0\\0&0&0&1&0&0&0&0\\0&0&1&0&0&0&0&0\\0&0&0&0&1&0&0&0\\0&0&0&0&0&1&0&0\\0&0&0&0&0&0&0&1\\0&0&0&0&0&0&1&0\end{pmatrix}\begin{pmatrix}w_0\\w_1\\w_2\\w_3\\w_4\\w_5\\w_6\\w_7\end{pmatrix}=\begin{pmatrix}w_0\\w_1\\w_3\\w_2\\w_4\\w_5\\w_7\\w_6\end{pmatrix}$$

矩阵 $\mathbf{Q}$ 定义如下：

$$\begin{aligned}&\mathbf{Q}_{k,2k}=\mathbf{Q}_{N/2+k,2k+1}=1 \quad k=0,\ldots,N/2-1\\&\mathbf{Q}_{k,l}=0 \qquad\qquad else\end{aligned} \tag{3—27}$$

经过置换矩阵 $\mathbf{Q}$ 向量的前半部分为原来偶下标的分量，接着是奇下标的分量，以 $\vec{w}=(w_0\ w_1\ w_2\ w_3\ w_4\ w_5\ w_6\ w_7)^T$ 为例，经过置换矩阵 $\mathbf{Q}$，得到

$$Q\vec{w}=\begin{pmatrix}1&0&0&0&0&0&0&0\\0&0&1&0&0&0&0&0\\0&0&0&0&1&0&0&0\\0&0&0&0&0&0&1&0\\0&1&0&0&0&0&0&0\\0&0&0&1&0&0&0&0\\0&0&0&0&0&1&0&0\\0&0&0&0&0&0&0&1\end{pmatrix}\begin{pmatrix}w_0\\w_1\\w_2\\w_3\\w_4\\w_5\\w_6\\w_7\end{pmatrix}=\begin{pmatrix}w_0\\w_2\\w_4\\w_6\\w_1\\w_3\\w_5\\w_7\end{pmatrix}$$

因此 N 点的 DCT_{IV} 可以分解为：

$$\mathbf{DCT}_N^{IV}=\mathbf{L}_N\begin{pmatrix}\mathbf{DCT}_{N/2}^{IV}&\mathbf{0}_{N/2}\\\mathbf{0}_{N/2}&\mathbf{DCT}_{N/2}^{IV}\end{pmatrix}\mathbf{M}_N\mathbf{Q}_N\mathbf{P}_N \tag{3—28}$$

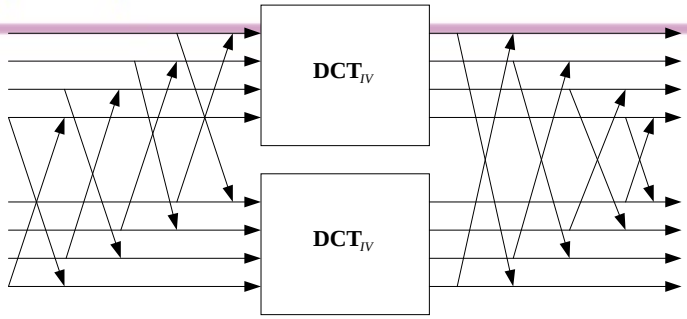


图 3-3 长度 N 的 DCT_{IV} 分解两个长度 N/2 的 DCT_{IV} 和 Givens 旋转

Figure 3-3 DCT_{IV} of length N by two DCT_{IV} of length N/2 and two stages of Givens rotations

则用式（3-13）长度为 N/2 的$\mathbf{DCT}_{N/2}^{IV}$可以分解为多维提升。两块长度 N/2 的$\mathbf{DCT}_{N/2}^{IV}$使用多维提升,如前所述需要 3N/2 次取整操作. 矩阵 L 和 M 可以分别使用 N/2 个 Givens 旋转实现，即分别需要 3N/2 次取整操作。这样总共需要 3N/2+2×3N/2=9N/2 次取整操作。

3.3 整数 MDCT

整数MDCT是MDCT的整数逼近，具有可逆性，可以通过提升方法来实现，并且保留了MDCT的大部分优点[12] 如具有好的频率选择性以及临界采样和块重叠性质。

3.3.1 MDCT 分解

对于块t，2N个时域样点$x_t(k)\quad k=0,...,2N-1$，计算出N个谱线$X_t(m)\quad m=0,...,N-1$，MDCT定义如下[15]：

$$X_t(m)=\sqrt{\frac{2}{N}}\sum_{k=0}^{2N-1}w(k)x_t(k)\cos\frac{(2k+1+N)(2m+1)\pi}{4N}\quad m=0,...,N-1 \qquad (3-29)$$

逆变换

$$y_t(k)=w(k)\sqrt{\frac{2}{N}}\sum_{m=0}^{N-1}X_t(m)\cos\frac{(2k+1+N)(2m+1)\pi}{4N}\quad k=0,...,2N-1 \qquad (3-30)$$

使用前向和后向 MDCT 将引入时域混叠误差。误差可以通过两个相邻块 t 和 t＋1 重叠部分的 MDCT 输出相加来消除：

$$x_t'(k)=y_t(N+k)+y_{t+1}(k)\qquad k=0,...,N-1 \qquad (3-31)$$

由于正交变换在边界处存在着固有的不连续性（只不过不同类型的正交变换，其不连续性的程度不同），因此在这些分组边界处就可能产生很大的噪声，即所谓的“边界效应”。MDCT 利用时域的重叠抵消技术来减小边界效应。对于输入序列$x(k)$用本组的 N 个样点和前后两个相邻组的各一半的样点重叠构成 2N

个样值，加 2N 点长的窗函数作变换。时域重叠抵消的一个充分条件为：

$$\left.\begin{aligned} w(k)^2 + w(N+k)^2 &= 1 \\ w(k) &= w(2N-1-k) \end{aligned}\right\} \qquad k=0,\ldots,\text{N-1} \qquad (3-32)$$

其中 w 即为窗函数，满足该条件的窗函数的一个例子是正弦窗

$$w(k) = \sin\left(\frac{\pi}{4N}(2k+1)\right) \quad k = 0,\ldots,2N-1$$

本文提出的无损音频压缩算法的整数 MDCT 中使用的即为正弦窗。

下面介绍将 2N 点 MDCT 变换分解为 Givens 旋转和 N 点 IV 型 DCT 变换。

令 $x(k) = w(k)x_t(k)$

因此 MDCT 定义式（3—29）化为：

$$\begin{aligned} X_t(m) &= \sqrt{\frac{2}{N}}\sum_{k=0}^{2N-1} x(k)\cos\frac{(2k+1+N)(2m+1)\pi}{4N} \\ &= \sqrt{\frac{2}{N}}\Big(\sum_{k=\frac{3N}{2}}^{2N-1} x(k)\cos\Big(\frac{\pi}{4N}(2m+1)(2k+1+N)\Big) \\ &\quad + \sum_{k=0}^{\frac{3N}{2}-1} x(k)\cos\Big(\frac{\pi}{4N}(2m+1)(2k+1+N)\Big)\Big) \end{aligned} \qquad (3-33)$$

$$m = 0,\ldots,N-1$$

对上式作变量替换

前半部分令 $k' = k - \frac{3N}{2}$

后半部分令 $k' = k + \frac{N}{2}$

$$\therefore \quad X_t(m) = \sqrt{\frac{2}{N}}\Big(\sum_{k=0}^{\frac{N}{2}-1} x\Big(k+\frac{3N}{2}\Big)\cos\Big(\frac{\pi}{4N}(2m+1)(2k+1+N+3N)\Big) + \sum_{k=\frac{N}{2}}^{2N-1} x\Big(k-\frac{N}{2}\Big)\cos\Big(\frac{\pi}{4N}(2m+1)(2k+1+N-N)\Big)\Big)$$

$$=\sqrt{\frac{2}{N}}\left(\sum_{k=0}^{\frac{N}{2}-1}x(k+\frac{3N}{2})\cos(\frac{\pi}{4N}(2m+1)(2k+1)+(2m+1)\pi)\right.$$
$$\left.+\sum_{k=\frac{N}{2}}^{2N-1}x(k-\frac{N}{2})\cos(\frac{\pi}{4N}(2m+1)(2k+1))\right)$$

$$=\sqrt{\frac{2}{N}}\left(\sum_{k=0}^{\frac{N}{2}-1}-x(k+\frac{3N}{2})\cos(\frac{\pi}{4N}(2m+1)(2k+1))\right.$$
$$\left.+\sum_{k=\frac{N}{2}}^{2N-1}x(k-\frac{N}{2})\cos(\frac{\pi}{4N}(2m+1)(2k+1))\right) \quad (3-34)$$

令 $y(k)=\begin{cases}-x(k+\frac{3N}{2}) & k=0,1,...,\frac{N}{2}-1\\ x(k-\frac{N}{2}) & k=\frac{N}{2},\frac{N}{2}-1,...,2N-1\end{cases}$ （3—35）

$$\therefore X(m)=\sum_{k=0}^{2N-1}y(k)\cos(\frac{\pi}{4N}(2m+1)(2k+1)) \qquad m=0,1,...,N-1 \quad (3-36)$$

$$\because \cos(\frac{\pi}{4N}(2m+1)(2k+1))=-\cos(\pi-\frac{\pi}{4N}(2m+1)(2k+1))$$
$$=-\cos(\frac{\pi}{4N}(4N-2k-1)(2m+1)) \quad (3-37)$$

$$\therefore \quad X(m)=\sum_{k=0}^{2N-1}y(k)\cos(\frac{\pi}{4N}(2m+1)(2k+1))$$
$$=\sum_{k=0}^{N-1}y(k)\cos(\frac{\pi}{4N}(2m+1)(2k+1))+\sum_{k=N}^{2N-1}y(k)\cos(\frac{\pi}{4N}(2m+1)(2k+1))$$
$$=\sum_{k=0}^{N-1}y(k)\cos(\frac{\pi}{4N}(2k+1)(2m+1))+$$
$$\sum_{k=0}^{N-1}y(2N-1-k)\cos(\frac{\pi}{4N}(4N-2k-1)(2m+1))$$
$$=\sum_{k=0}^{N-1}y(k)\cos(\frac{\pi}{4N}(2k+1)(2m+1))-$$
$$\sum_{k=0}^{N-1}y(2N-1-k)\cos(\frac{\pi}{4N}(2k+1)(2m+1))$$
$$=\sum_{k=0}^{N-1}(y(k)-y(2N-1-k))\cos(\frac{\pi}{4N}(2k+1)(2m+1))$$

（ 3 —38）

由式（3—34） $y(k)=\begin{cases}-x(k+\frac{3N}{2}) & k=0,1,...,\frac{N}{2}-1\\ x(k-\frac{N}{2}) & k=\frac{N}{2},\frac{N}{2}-1,...,2N-1\end{cases}$

$$\therefore y(2N-1-k)=\begin{cases}-x(\frac{7N}{2}-1-k) & k=\frac{3N}{2},...,2N-1\\ x(\frac{3N}{2}-1-k) & k=0,1,...,\frac{3N}{2}-1\end{cases} \quad (3-39)$$

$\because k\in[0,N-1]\ \therefore y(2N-1-k)=x(\frac{3N}{2}-1-k)$

$$\begin{aligned}\therefore X(m)&=\sum_{k=0}^{N-1}(y(k)-y(2N-1-k))\cos(\frac{\pi}{4N}(2k+1)(2m+1))\\ &=\sum_{k=0}^{\frac{N}{2}-1}(-x(k+\frac{3N}{2})-x(\frac{3N}{2}-1-k))\cos(\frac{\pi}{4N}(2k+1)(2m+1))\\ &\quad+\sum_{k=\frac{N}{2}}^{N-1}(x(k-\frac{N}{2})-x(\frac{3N}{2}-1-k))\cos(\frac{\pi}{4N}(2k+1)(2m+1))\end{aligned} \quad (3-40)$$

令 $\tilde{x}_t(N-1-k)=\begin{cases}x(\frac{3N}{2}+k)+x(\frac{3N}{2}-1-k) & k=0,1,...,\frac{N}{2}-1\\ -x(k-\frac{N}{2})+x(\frac{3N}{2}-1-k) & k=\frac{N}{2},...,N-1\end{cases}$ （3—41）

$$\therefore X(m)=\sum_{k=0}^{N-1}(-\tilde{x}_t(N-1-k))\cos(\frac{\pi}{4N}(2k+1)(2m+1))$$

对式（3—41）第二个式子作变量替换 $k'=N-1-k$

得 $\begin{cases}\tilde{x}_t(N-1-k)=x(\frac{3N}{2}+k)+x(\frac{3N}{2}-1-k)\\ \tilde{x}_t(k)=x(\frac{N}{2}+k)-x(\frac{N}{2}-1-k)\end{cases}$ $k=0,...,\frac{N}{2}-1$

又 $x(k)=w(k)x_t(k)$

即令 N 点信号 $\tilde{x}_t(k)$ 为

$$\begin{cases} \tilde{x}_t(k) = w(\frac{N}{2}+k)x_t(\frac{N}{2}+k) - w(\frac{N}{2}-1-k)x_t(\frac{N}{2}-1-k) \\ \tilde{x}_t(N-1-k) = w(\frac{3N}{2}+k)x_t(\frac{3N}{2}+k) + w(\frac{3N}{2}-1-k)x_t(\frac{3N}{2}-1-k) \end{cases}$$

$$k = 0,\ldots,\frac{N}{2}-1 \qquad (3-42)$$

对信号 $x_t(k)$ 作 2N 点的 MDCT 变换可转化为信号 $\tilde{x}_t(N-1-k)$ 作 N 点的 DCT 变换

$$X_t(m) = \sqrt{\frac{2}{N}}\sum_{k=0}^{N-1}\tilde{x}_t(N-1-k)\cos(\frac{\pi}{4N}(2k+1)(2m+1)) \qquad (3-43)$$

另，由式（3－42）用于t－1块MDCT的输入信号

$$\tilde{x}_{t-1}(N-1-k) = w(\frac{3N}{2}+k)x_{t-1}(\frac{3N}{2}+k) + w(\frac{3N}{2}-1-k)x_{t-1}(\frac{3N}{2}-1-k)$$

又块 t 与块 t－1 有 N 个样点重合

$$x_{t-1}(\frac{3N}{2}+k) = x_t(\frac{N}{2}+k)$$

$$x_{t-1}(\frac{3N}{2}-1-k) = x_t(\frac{N}{2}-1-k)$$

又由窗的性质式（3－32）

$$w(k) = w(2N-1-k)$$

$$\therefore \quad \tilde{x}_{t-1}(N-1-k) = w(\frac{3N}{2}+k)x_t(\frac{N}{2}+k) + w(\frac{3N}{2}-1-k)x_t(\frac{N}{2}-1-k)$$

$$= w(\frac{N}{2}-1-k)x_t(\frac{N}{2}+k) + w(\frac{N}{2}+k)x_t(\frac{N}{2}-1-k)$$

$$k = 0,\ldots,\frac{N}{2}-1$$

又结合式（3－42），块 t 的

$$\begin{pmatrix} \tilde{x}_t(k) \\ \tilde{x}_{t-1}(N-1-k) \end{pmatrix} = \begin{pmatrix} w(\frac{N}{2}+k) & -w(\frac{N}{2}-1-k) \\ w(\frac{N}{2}-1-k) & w(\frac{N}{2}+k) \end{pmatrix} \begin{pmatrix} x_t(\frac{N}{2}+k) \\ x_t(\frac{N}{2}-1-k) \end{pmatrix}$$

$$k = 0,\ldots,\frac{N}{2}-1$$

$$(3-44)$$

由时域重叠抵消的条件[7]，上式中变换矩阵的行列式为1，因此实际上它是Givens

旋转。

图3-4描述了上述由N点DCT_{IV}实现2N点MDCT的具体过程

3.3.2 整数 MDCT 的实现

上述分解整个MDCT变换拆分为加窗、时域重叠抵消和长度为N的IV型离散余弦变换（DCT_{IV}）等三个步骤，由时域重叠抵消条件可知，加窗、时域重叠抵消步骤可以由Givens旋转实现。即MDCT分解为Givens和IV型离散余弦变换（DCT_{IV}）。由第一节给出的Givens旋转的整数近似和第二节介绍的采用经典提升和多维提升实现整数DCT，可以得到两种整数MDCT变换的实现。

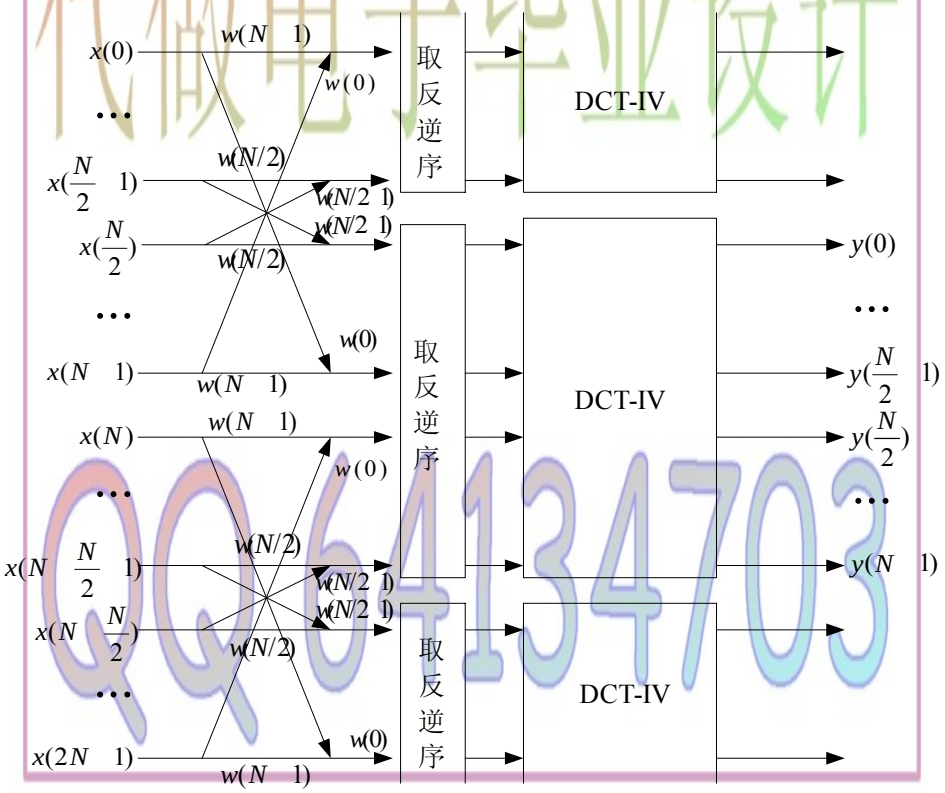


图 3－4　MDCT 为 Givens 旋转和 DCT－IV

Figure 3-4 MDCT and inverse MDCT by Givens rotations and DCT_{IV}

图3－5给出了SQAM[6]（用于主观测试的音乐库）中测试曲目Track64的IntMDCT和MDCT幅度谱以及逼近误差。由图可以看到，整数MDCT变换后的结果与浮点变换结果相差无几，可以有效去除信号的相关性，良好地表示信号的谱信息。

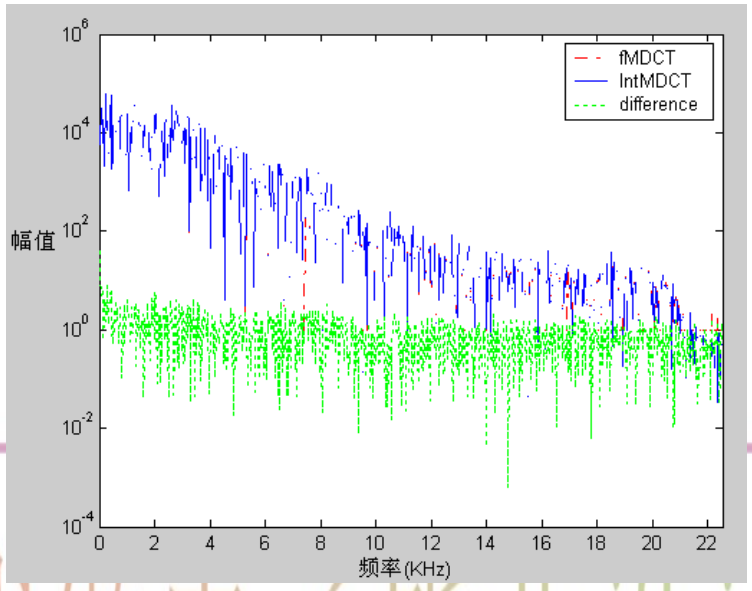


图3－5　SQAM[6]中测试曲目Track64的IntMDCT和MDCT幅度谱以及逼近误差

Figure 3-5　IntMDCT and MDCT magnitude spectra and difference values of track 64, SQAM[6]

3.4 本章小结

本章主要介绍了整数 IV 型 DCT 和整数 MDCT 的实现算法。依据矩阵分解的理论，本文证明 MDCT 分解为 Givens 旋转和 DCT 变换的推导过程，并在此基础上采用“经典提升” 和“多维提升”方案，实现整数改进型离散余弦变换（IntMDCT），并且提出了实际编程中的简化算法。整数 MDCT 的充分研究，为将其应用于无损音频压缩做好准备。

第四章　基于 IntMDCT 的无损音频压缩算法设计

本章提出一种基于IntMDCT的无损音频压缩算法。由于在整数变换的过程中，使得信号的能量分布更为集中，为熵编码提供了有利条件，因而可以实现较为有效的压缩。变换后的系数经过Golomb-Rice编码并形成码流输出。由于该算法完全可逆，则解码运算是编码运算的逆流程。本章首先详细描述基于IntMDCT无损音频压缩的编码算法；接着描述相应的解码算法，最后给出适合本文算法的比特流结构设计。

4.1 编码算法

根据图2—2独立的无损音频压缩编码方案，图4—1给出了本文设计的音频无损压缩编解码方案。音频信号首先经过分帧处理。在一帧时间内将音频信号作为短时平稳信号进行处理，并且方便对压缩后的比特流进行编辑以及实现音频信息的随机访问。分帧后的信号送入IntMDCT变换模块进行整数变换。变换后的系数进行映射后送入Golomb-Rice编码器。系数的编码值结合子带划分信息、编码参数等边信息形成压缩码流输出。解码端的操作相反，先进行Golomb-Rice解码，然后作IntMDCT逆变换，恢复出原始数据。由于在整数变换的过程中，使得信号的能量分布更为集中，为熵编码提供了有利条件，因而可以实现较为有效的压缩。

整数MDCT模块的设计已在第三章中详细介绍。下面重点介绍编码算法中其他重要模块的设计。

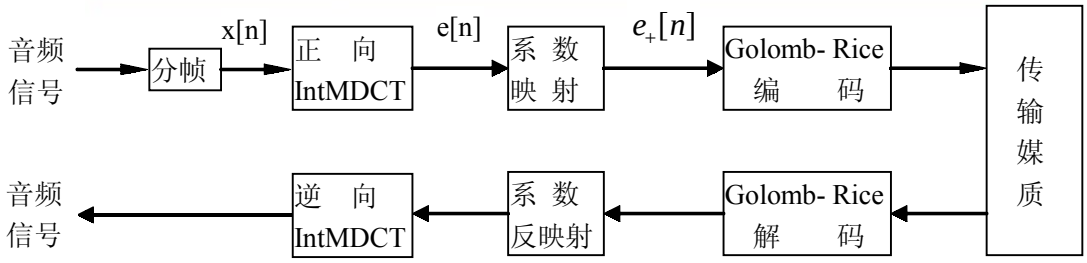


图 4—1 无损压缩编解码方案

Figure 4-1 Lossless audio compression codec scheme

4.1.1 Golomb-Rice 编码器

经过IntMDCT变换后的系数均为整数，将其送入熵编码模块。Golomb-Rice编码用于消除变换后系数的冗余度，它本身并不引入失真。Golomb-Rice码是一种特殊的Huffman码，适用于几何分布的正整数信号。为了便于编码，希望对经过IntMDCT变换后的系数进行映射，从而使得映射后的系数都为正整数。本文采用的映射公式如下[16]：

$$e_+[n]=\begin{cases}2e[n] & e[n]\geq 0\\ 2|e[n]|-1 & otherwise\end{cases} \quad (4-1)$$

本文对映射后系数的统计特性进行了估计，如图4—2虚线所示，该图由大量音频信号变换系数的统计结果得到。图中同时给出了几何分布的理论曲线 $f(n)=p(1-p)^n$（图4—5中的实线），其中参数 p 为0.038。从图中可以看出，经过映射后的IntMDCT系数的概率分布接近于几何分布，因此采用Golomb-Rice码可以达到较好的编码性能。

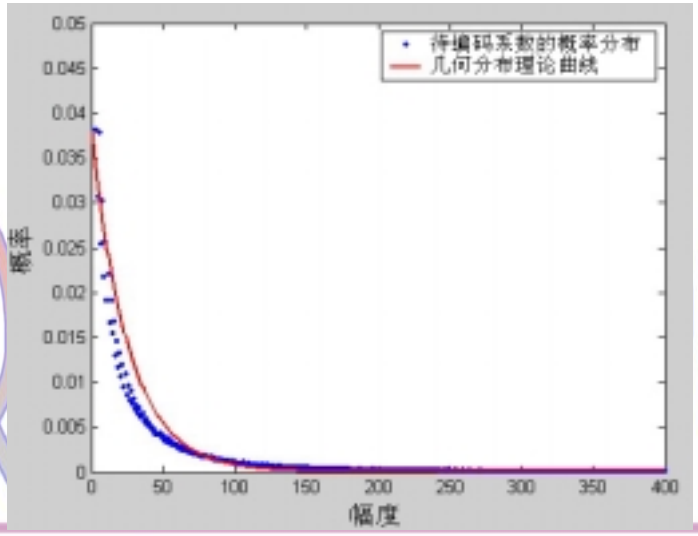


图4-2 几何分布（实线）与待编码系数的概率分布（虚线）

Figure 4-2 Geometry distributions(real line) and probability distributions of coefficient (dashed)

Golomb-Rice码有一个参数s，对于正整数n，编码分成三部分，(n>>s)个比特“0”，一个比特“1”作为分割符，n二进制表示的低s比特。例如：s=2，n=13的编码为：（13>>2）等于3个比特“0”，一个比特“1”，13二进制表示的低2比特为“01”，即000101。

参数s的选取需要利用变换后系数的统计特性，一种估计的方法是：

$$s=\log_2(\log_e(2)E(|e_+(n)|)) \quad (4-2)$$

其中，E[g]表示数学期望

根据数值的范围选择合适的编码参数s，对于压缩效率有重要的影响。例如需要对1000进行编码，表4－1给出对于不同的编码参数所使用的编码比特数

表4－1 不同GR编码参数情况使用的比特数

Table 4-1 Codeword lengths of different GR parameter

s=1	s=2	s=3	s=4	s=5	s=6	s=7	s=8	s=9	s=10	s=11	s=12
502	253	129	67	37	22	15	12	11	11	12	13

由上表可以看出，使用参数s＝9或者10，可以使用最少的比特数表示样值1000。如果选择的参数不合适，将造成编码比特数的浪费。因此选择合适的编码参数对于整个压缩算法具有重要的意义。

由于音频信号谱在高低频带呈现出不同的幅度特性，如图4－3所示，从图中可以看到，变换后低频部分（0－4KHz）系数的数值偏大，数值浮动范围大；中频部分（4－8KHz）数值中等，比较平稳；高频部分（8KHz以上）数值普遍较小。因此为了提高编码性能，本文根据信号统计特征的差异，不同的频段内采用不同的编码参数。

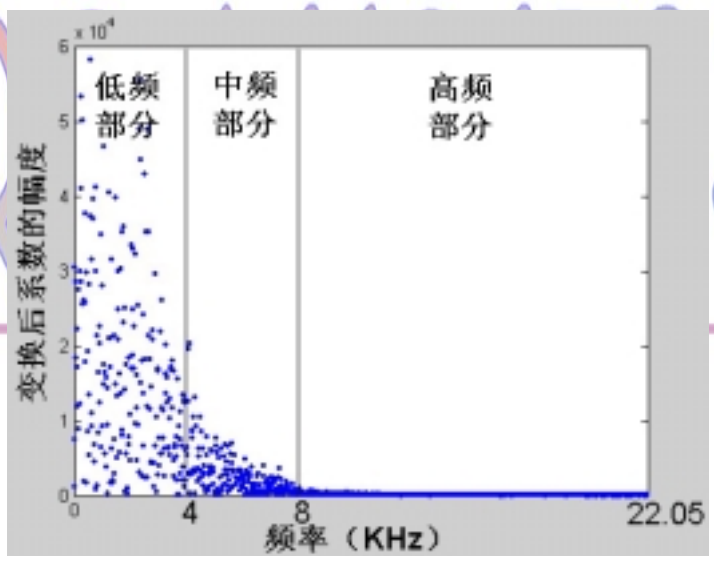


图 4-3　SQAM[6]测试曲目 Track66 的一帧音频数据的 IntMDCT 谱

Figure 4-3 Absolute values of IntMDCT spectrum ,length 1024 , Track66 in SQAM

4.1.2 CRC 检验

从上述编码算法可以看到，算法中一些关键信息如子带数目信息、子带划分信息、GR 编码参数信息对于解码有着重大影响，由于传输存储等错误造成关键信息的读取错误，将会导致整个音频帧的解码错误。因此考虑在编码中对这些关键信息作一些保护。使用的错误检测方法是国际上常用的循环冗余校验码“CRC－16”。如果编码算法中使用 CRC 校验，头信息中保护位等于“0”，并且在比特流中紧随头信息后插入了一个 CRC 校验字。

CRC－16 发生器多项式为：

$$G(X)=X^{16}+X^{15}+X^{2}+1$$

图 4－4“CRC 校验图”描绘了此方法。移位寄存器的初始状态为“1111 1111 1111 1111”，编码时将所有需要进行 CRC－校验的比特输入到图 4－4“CRC 校验图”中所示的电路中。每输入一比特，移位寄存器移动一位。在最后一个移位操作完成后，输出 b15…b0 组成一字。将这个校验字写入比特流中。解码时，读出关键信息，同样进行 CRC 校验后得到校验字，与比特流中的 CRC 校验字比较。如果它们不同，则比特流的保护的关键信息出错。

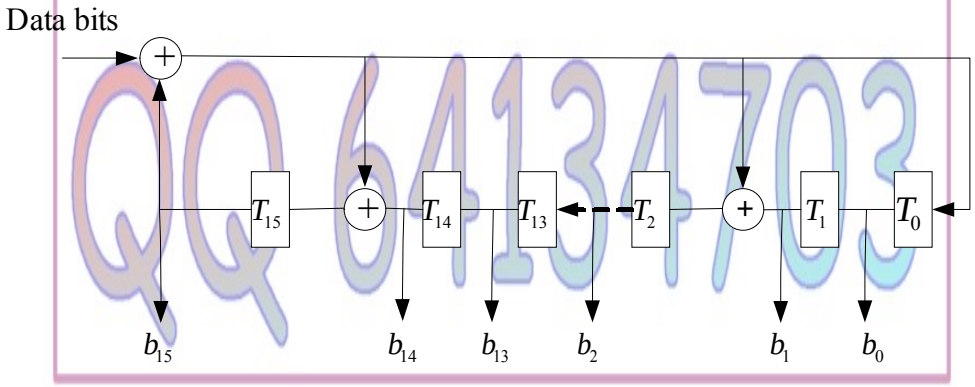


图 4-4 CRC 校验图

Figure 4-4 CRC check

使用 CRC 校验提高了算法的鲁棒性。在检测到传输的重要信息出错时，使用重复前一帧或者直接跳过该帧，避免由于解码错误造成的音频失真。

4.1.3 立体声编码

以上算法设计中只是利用了声道内音频信号的相关性进行压缩，还可以考

虑利用声道间的相关性提高编码效率即立体声编码，本算法可以考虑采用 M/S 立体声编码，将左右声道的信息转化为左右声道的中点（mid）和左右声道之差（side）来编码，对于两个声道相似度较大的情况，立体声编码方式能比较有效地提高编码效率。

选用参数为$\frac{\pi}{4}$的 Givens 旋转可以实现无损的 M/S 立体声编码。

$$\begin{pmatrix} \cos\frac{\pi}{4} & -\sin\frac{\pi}{4} \\ \sin\frac{\pi}{4} & \cos\frac{\pi}{4} \end{pmatrix} = \frac{\sqrt{2}}{2}\begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}$$

4.2 解码算法

由于该算法完全可逆，则解码运算是编码运算的逆流程。解码时首先在码流中搜寻 12 位同步字“1111 1111 1111”，搜索到同步字后，读出并分析头信息。根据头信息，确定音频的采样频率、字长和声道数。根据头信息中的帧长信息，可以确定下一个同步字的位置，可以实现音频的随即访问。如果头信息中保护位等于“0”，在头信息后读出一个 16 比特的 CRC 校验字。读出子带数目信息、子带划分信息和参数信息。对关键信息作 CRC 校验并与读出的校验字相比较，检查关键信息是否出错。根据子带划分信息计算每个子带内含有多少个样点，由参数信息确定子带内每个样点 Golomb-Rice 的编码参数 s。读出样点信息，根据得到的参数 s，进行 Golomb-Rice 解码。对于每个编码值，读出 m 个比特“0”，一个比特“1”，s 比特的数值 d，所以经过 GR 解码的样点值为 $n=m\times 2^s+d$。然后对解码的结果作反映射

$$e[n]=\begin{cases} e_+[n]/2 & e_+[n]\text{为偶数} \\ (e_+[n]+1)/2 & e_+[n]\text{为奇数} \end{cases} \quad (4-3)$$

对映射后的系数作整数 MDCT 逆变换，恢复出原始的音频数据。

4.3 比特流结构设计

音频帧组织如下：头信息，错误校验，音频数据和附加信息。音频数据包

括边信息（子带数目信息，子带分配信息和参数信息）以及样值信息。

同步头（40 比特）	CRC（16 比特）	子带数目 K（5 比特）	子带分配信息（K×5 比特）	参数信息（K×4 比特）	样本信息	附加数据

图 4-5　基于 IntMDCT 无损音频编码算法音频数据格式（一帧）

Figure 4-5　Bit stream syntax of lossless audio compression algorithm based on IntMDCT

为了方便说明，下面用伪代码表示比特流结构的设置。

4.3.1 音频序列

```
audio sequence()
{
    while (nextbits() = = syncword )
        frame()
}
```

4.3.2 音频帧

音频帧：含有 1024 个样点信息，它起始于一个同步字，终止于下一个同步字，每一帧由整数个字节组成，字节中未使用的多余比特用 0 或 1 填充。

```
frame()
{
    header()
    error－check()
    audio_data()
    ancillary-data()
}
```

4.3.3 头信息

```
header()
{
    syncword                    16bit
```

version	2 bit
frame_len	13bit
protection-bit	1 bit
sampling-frequency	3 bit
mode	2 bit
word-length	3 bit

}

所有帧的开始 40bit（5 字节）都是头信息，包含同步和一些状态信息：

同步字（syncword）：16bit，比特串“0111 1111 1111 1111”作为每一帧的开始

版本号（version）：2bit，指明使用哪个版本（算法扩展用）

帧长（frame_len）：13bit，给出本帧所用的字节数，即可确定下一同步头的位置，便于实现音频的随机访问

保护一位（protection-bit）：1bit，用来指明编码算法中是否对关键信息作 CRC 检验。等于“1”表示无校验；等于“0”表示加入了校验信息

采样频率（sampling-frequency）：3bit，指明音频的采样频率

	频率(kHz)
‘000’	44.1
‘001’	48
‘010’	32
‘011’	192
‘100’	96
‘101’ ‘110’ ‘111’	保留

字长（word-length）：3bit，指明原始音频的量化比特数

	字长(bit)
‘000’	16
‘001’	20
‘010’	24
‘011’	8
‘100’ ‘101’ ‘110’	保留

‘111’	

模式（mode）：2bit，指明编码的模式

	模式
‘00’	立体声
‘01’	联合立体声(强度立体声或者 M/S 立体声)
‘10’	双声道
‘11’	单声道

4.3.4 CRC 校验信息

错误一检查(error-check)：16bit，包含 CRC 校验信息。被保护的音频数据比特域：头信息的第 31 到 39 比特，子带数目信息，子带划分信息，GR 参数信息。

```
error-check()
{
    if (protection-bit = = 0)
        crc-check                16bit
}
```

4.3.5 音频数据

```
audio-data()
{
    for (ch=0; ch<nch; ch++)
    {
        subband-number[ch]                    5bit
        for (sb=0; sb<subband-number; sb++)
        {
            subband-division[ch][sb]          5bit
            parameter[ch][sb]                 4bit
        }
    }
```

```
        for (ch=0; ch<nch; ch++)
            for (s=0; s<M; samp++)
            {
                sample[ch][s]
            }
}
```

音频数据包含子带数目信息、子带划分信息、GR 参数信息、音频样点信息。

子带数目信息（subband-number[ch]）：一帧样点划分为子带个数

子带划分信息（subband-division[ch][sb]）：给出的信息是关于 ch 声道第 sb 个子带的划分，32 个样点为一个基带，信息给出 ch 声道第 sb 个子带中含有几个基带。

参数信息（paremeter[ch][sb]）：ch 声道第 sb 个子带的 Golomb-Rice 的参数信息

样点信息（sample[ch][s]）：　　ch 声道第 s 个样点的 Golomb-Rice 的编码表示

根据子带划分信息和参数信息，得出ch声道第s个样点的Golomb-Rice的参数

4.3.6 辅助数据

辅助数据(ancillary-data)：用于辅助数据和多声道扩展

4.4 本章小结

本章将整数 MDCT 应用于无损音频压缩，详细描述音频信号经过整数 MDCT 变换后，根据变换系数高低频段的不同幅度分布特性划分子带，每个子带中采用相同的 GR 编码参数进行 Golomb-Rice 编码，将子带数目信息、子带划分信息、编码参数信息和编码样点信息形成压缩码流后输出。给出相应的解码算法，并提出适合该编解码算法的比特流结构。

第五章　算法实现和性能分析

这一章首先给出本文无损音频编解码算法的具体实现，在此基础上，根据软件仿真的结果，分析算法性能如压缩效率、算法鲁棒性以及算法复杂度分析，并与目前典型的无损音频算法进行比较与分析。

5.1 算法实现

编码时，首先读入wav文件，分析wav文件的头信息，得到原始音频的采样频率、字长和声道数目。根据这些信息设置压缩码流的头信息中的一些状态位。分帧后的信号送入IntMDCT变换模块进行整数变换。变换后的系数送入Golomb-Rice编码器。系数编码后的码字结合子带划分信息、编码参数等边信息形成压缩码流输出。下面给出算法实现时需要考虑的一些问题。

5.1.1 分帧长度的确定

进行整数变换前，需对音频作分帧操作，以便于数字化音频处理。由前所述，分帧长度对于算法的性能有很大影响，下面通过仿真实验确定算法中采用的帧长。表 5—1 中是不同采样率的情况下，采用不同分帧长度得到的压缩比比较。测试曲目是 SQAM（Sound Quality Assessment Material –用于主观测试的音乐库）中 Track69，采样率 44.1kHz。通过升采样和降采样，得到采样率为 32kHz 和 48kHz 的音频用于测试。

通过表 5—1，可以看到，采用帧长 1024 可以得到最大的压缩比。下面的仿真算法中采用的都是 1024 点帧长。

5.1.2 模式的选择

由设计的比特流结构，编码模式可以考虑采用单声道、双声道和立体声编码等方式。若原始音频为双声道数据，可以利用信道间的相关性进一步提高压缩比即采用立体声编码。表 5—2 给出了采用双声道与立体声两种编码方式得到

的压缩效果。Track69 是 SQAM 中的测试曲目，Track L=R 是由 Track69 的右声道构造，左右声道数据完全相同。表中第一列是原文件的大小，第二列是经过双声道模式编码后的文件大小，第二列是经过立体声模式编码后的文件大小，单位：字节。可以看到对于 Track69 两种方式的压缩比相差无几。而对于左右声道相关性很大的 Track L=R，立体声编码的方式明显优于双声道模式。

一般来说，两声道数据虽不是完全独立，但由于其相关性较弱，难以用于音频压缩。且采用立体声编码后，计算复杂度大大增加，每两个样点多 3 次取整操作，即平均每个样点多 1.5 次取整操作。因此下面的测试都是未使用立体声编码模式。

表 5－1 不同采样率下的帧长选择

Table 5-1 Selection of frame length for various sampling frequency

采样率（kHz）	选用帧长	原文件大小（字节）	压缩后文件大小（字节）	压缩比
44.1	512	4,987,124	2,414,262	2.0656
	1024		2,378,672	2.0966
	2048		2,413,383	2.0664
32	512	3,618,716	1,914,390	1.8903
	1024		1,903,111	1.9015
	2048		1,950,603	1.8552
48	512	5,428,160	2,509,058	2.1634
	1024		2,470,644	2.1971
	2048		2,496,520	2.1743

表 5－2 双声道与立体声编码压缩结果

Table 5-2 Compression result of dual and stereo mode

	原文件大小（字节）	双声道模式（字节）	立体声模式（字节）
Track L=R	4,987,124	2,367,348	1,645,435
Track 69	4,987,124	2,378,672	2,378,363

5.1.3 整数 MDCT 的实现

第三章介绍了两种提升的方法——经典提升和多维提升。基于这两种不同提升，相应有两种实现整数MDCT的方法。表5－3将基于两种提升方法实现整数MDCT的方法作一比较。表中第二列是基于经典提升实现整数MDCT的各种指标，第三列是基于多维提升实现整数MDCT的各种指标，第四列是对浮点MDCT变换系数直接取整后的结果的各种指标，它不是无损运算。比较的内容包括：平均每个样点的取整次数、指令数（加法和乘法的次数和）、均方误差（与浮点MDCT变换系数直接取整后的结果相比）和最大误差的绝对值。

无损音频压缩算法的整数变换模块分别采用基于经典提升的整数 MDCT 和基于多维提升的整数 MDCT，仿真结果表明，在帧长为 1024、未采用立体声编码模式情况下，后者比前者算法平均每帧节省约 400bit。SQAM 中的 Track69 原文件大小 4,987,124 字节，经过基于经典提升的算法压缩后，压缩文件大小为 2,493,562 字节；经过基于多维提升的算法压缩后，压缩文件大小为 2,378,672 字节。因此实现时整数 MDCT 时采用基于多维提升的整数变换算法。

表5－3 两种提升方法实现整数MDCT的比较

Table 5-3 Comparison of two lifting methods for IntMDCT implementation

	经典提升	多维提升	直接取整（有损）
每个样点取整次数	22.5	4	1
每个样点指令数	45	32	20
均方误差	1.97	0.48	0
最大误差的绝对值	8	4	0

5.1.4 子带划分

由第四章算法设计中所分析，不同频段内的信号统计特征有很大区别，因此不同频段内采用不同的编码参数以提高编码性能。本文将整个频带均分为32个子带，计算每个子带内样点的数学期望，由式(4-2)计算每个子带的编码参数。在中高频部分数据变化平稳，合并具有相同参数的相邻子带以节约编码参数所用比特，一般3、4个子带公用一个编码参数；低频部分，数据变化剧烈，每个

子带使用一个相应的编码参数。实验结果表明子带划分合并后一般都只传12个子带左右的参数，每个参数使用4bit编码，以较少的比特传送了合适的编码参数，得到了较高的压缩效率。

5.1.5 CRC 校验的实现

若对关键信息采用CRC校验，CRC校验的实现可以采用查表法，由于一个字节是8bit，所以每次处理一个字节，共有2^8＝256项。通过查表法进行运算，大大提高了处理速度，为故大多数应用所采用。本算法中使用的CRC校验表格见附录。

图 5－1 和 5－2 给出编解码算法的流程图。

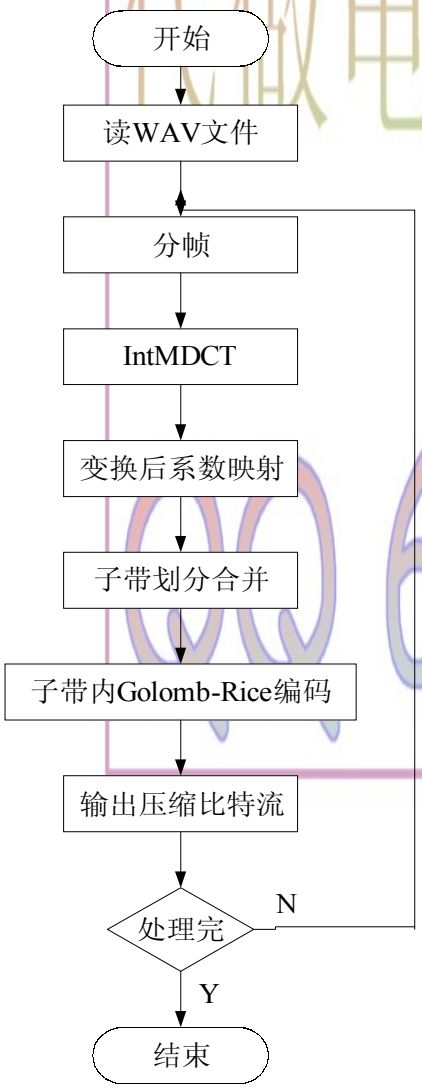


图 5－1 无损音频编码流程图

Fig 5-1 Flow chart of lossless audio encoder

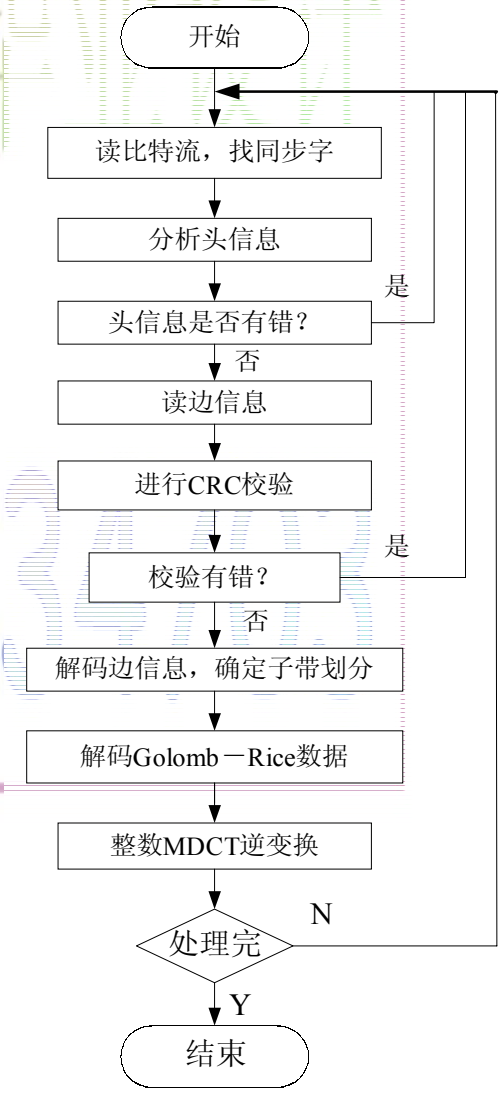


图 5－2 无损音频解码流程图

Fig 5-2 Flow chart of lossless audio decoder

5.2 实验结果

首先定义算法压缩比以及压缩后曲目的平均比特数为

$$压缩比=\frac{原文件大小}{压缩后文件大小}$$

$$平均比特数=\frac{压缩后文件大小}{原文件大小}\times 16$$

5.2.1 算法的压缩效率

本文提出的算法采用帧长 1024，即约 23ms 一帧进行整数变换。并且未使用联合立体声编码。实验数据为 SQAM[6]（Sound Quality Assessment Material – 用于主观测试的音乐库）中多种类型的音频信号，采样率 44.1KHz，每个采样值用 16bit 量化。

表5－4给出SQAM中测试曲目经过本文算法压缩后的结果。表中第1列是SQAM中各个曲目的名称，第2列给出每个曲目音频信号的描述，第3列列出了测试曲目的文件大小，单位兆字节（M Bytes），第4列测试曲目的持续时间，单位秒（sec），第5列列出经过本算法压缩后的文件大小，单位兆字节（M Bytes），第6、7列给出该曲目的压缩比和平均比特数。

表 5－4 仿真结果

Table 5-4 Simulation result

曲目	描述	原文件大小（M bytes）	持续时间（sec）	压缩文件大小（M bytes）	压缩比	平均比特数（bit）
Track01	正弦波（1KHz）	16.8	99.91	3.75	4.48	3.57
Track02	带限色噪声	7.66	45.568	3.69	2.07	7.70
Track03	合成电子锣（100Hz）	3.72	22.163	0.64	5.77	2.76
Track04	合成电子锣	4.26	25.355	0.704	6.05	2.64

	（400Hz）					
Track05	合成电子锣（5KHz）	3.75	22.331	0.724	5.17	3.08
Track06	合成电子锣（500Hz）	3.15	18.763	0.454	6.94	2.30
Track07	合成信号	5.25	31.219	1.03	5.09	3.13
Track08	乐器（小提琴）	11.3	67.402	5.11	2.21	7.23
Track09	乐器（中提琴）	8.81	52.393	3.96	2.22	7.19
Track10	乐器（大提琴）	11.9	71.223	4.47	2.66	6.01
Track11	乐器（低音提琴）	10.4	62.055	3.67	2.83	5.64
Track12	乐器（短笛）	3.96	23.566	1.48	2.67	5.97
Track13	乐器（长笛）	7.02	41.781	2.52	2.78	5.74
Track14	乐器（双簧管）	7.55	44.935	2.99	2.52	6.33
Track15	乐器（喇叭）	3.41	20.309	1.45	2.35	6.80
Track16	乐器（单簧管）	5.88	34.998	2.1	2.8	5.71
Track17	乐器（低音单簧管）	3.43	20.443	1.45	2.36	6.76
Track18	乐器（巴颂管）	5.65	33.605	1.91	2.95	5.40
Track19	乐器（低音巴颂管）	3.17	18.890	1.17	2.71	5.90
Track20	乐器（萨克斯管）	5.95	35.386	2.1	2.83	5.64
Track21	乐器（小号）	5.39	32.088	1.57	3.43	4.66
Track22	乐器（长号）	6.01	35.751	1.54	3.90	4.09
Track23	乐器（圆号）	6.76	40.214	1.68	4.03	3.97
Track24	乐器（大号）	7.56	44.986	1.67	4.52	3.53
Track25	乐器（竖琴）	10.7	64.170	2.75	3.89	4.11
Track26	乐器（克拉维	6.07	36.058	1.44	4.22	3.79

	棍）					
Track27	乐器（响板）	2.67	15.890	1.28	2.08	7.67
Track28	乐器（小鼓）	12.54	74.541	3.71	3.38	4.73
Track29	乐器（大鼓）	3.4	20.189	0.658	5.16	3.09
Track30	乐器（定音鼓）	3.93	23.358	0.8	4.91	3.25
Track31	乐器（钹）	20.22	120.214	5.82	3.47	4.60
Track32	乐器（铁三角）	11.9	70.758	3.59	3.31	4.82
Track33	乐器（锣）	44.29	263.302	10	4.43	3.61
Track34	乐器（管钟）	3.85	16.099	1.1	3.50	4.57
Track35	乐器（钟琴）	9.24	54.918	2.35	3.93	4.06
Track36	乐器（木琴）	7.76	46.109	2.19	3.54	4.51
Track37	乐器（颤音琴）	1.99	11.815	0.55	3.61	4.42
Track38	乐器（玛林巴琴）	2.94	17.446	0.563	5.22	3.06
Track39	乐器（三角钢琴）	22.55	134.018	5.99	3.76	4.25
Track40	乐器（大键琴）	8.27	49.166	3.05	2.71	5.90
Track41	乐器（钢片琴）	10.65	63.313	2.72	3.92	4.08
Track42	乐器（手风琴）	3.23	19.178	1.33	2.42	6.58
Track43	乐器（管风琴）	14.4	85.673	6.48	2.22	7.2
Track44	女高音	4.42	26.286	1.59	2.78	5.75
Track45	女低音	4.49	26.712	1.69	2.66	6.02
Track46	男高音	4.28	25.459	1.55	2.76	5.79
Track47	男低音	4.69	27.866	1.69	2.78	5.76
Track48	四重奏	4.15	24.688	1.74	2.39	6.70
Track49	女声语音，英语	3.27	19.430	1.5	2.18	7.33
Track50	男声语音，英语	3.61	21.457	1.45	2.49	6.42
Track51	女声语音，法语	2.95	17.559	1.47	2.49	7.97
Track52	男声语音，法语	3.44	20.430	1.52	2.26	7.06

Track53	女声语音，德语	2.81	16.696	1.28	2.20	7.28
Track54	男声语音，德语	2.84	16.894	1.28	2.21	7.21
Track55	独奏（小号）	4.86	28.906	1.79	2.71	5.89
Track56	独奏（管风琴）	5.12	30.406	2.56	2.00	8.00
Track57	独奏（管风琴）	2.71	16.109	1.15	2.35	6.78
Track58	独奏（吉他）	2.02	12.011	0.858	2.35	6.79
Track59	独奏（小提琴）	4.21	25.007	1.77	2.37	6.72
Track60	独奏（钢琴）	14.88	88.479	3.7	4.02	3.97
Track61	声乐（女高音）	29.51	295.394	12.1	2.43	6.56
Track62	声乐（女高音）	4.63	27.508	1.58	2.93	5.46
Track63	声乐（独唱）	8.92	53.049	3.84	2.32	6.88
Track64	声乐（合唱）	4.56	27.087	2.54	1.80	8.91
Track65	管弦乐	18.23	108.388	7.14	2.55	6.26
Track66	管弦乐（合唱）	2.36	14.057	1.16	2.03	7.86
Track67	管弦乐（合唱）	13.1	77.882	4.69	2.79	5.72
Track68	管弦乐	27	280.489	8.22	3.28	4.87
Track69	音乐	4.76	28.271	2.26	2.11	7.59
Track70	音乐	2.71	16.099	1.36	1.99	8.02
平均					3.17	5.57

对于不同的音频曲目，本算法的压缩比相差很大。例如，表 5-1 中对于 Track60 压缩比可以达到 4.02，而 Track64 的压缩比只有 1.8。这是由于 Track64 中含有合唱和交响乐类型的音频，信号变化剧烈，高频分量十分丰富，如图 5-3 中左图所示，因此很难压缩。相反，对于 Track60，音频信号比较平稳，高频分量很少，整数变换后信号能量集中于低频部分，去相关效果很好，容易压缩。

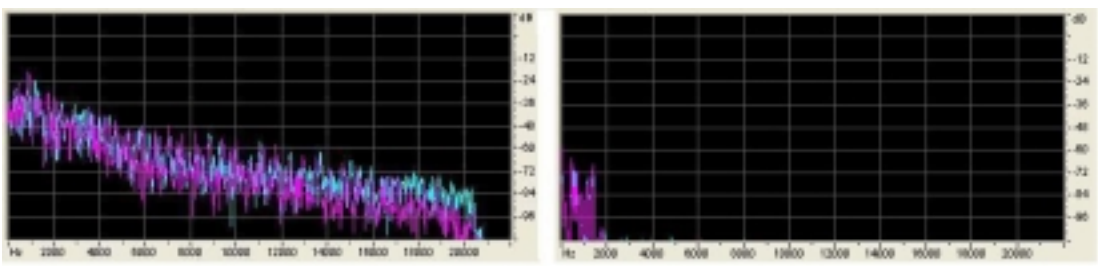


图 5—3 Track64 和 Track60 的频谱分析

Figure 5-3 the specturm of Track64 and Track60

从表中对于 SQAM 中 70 个测试曲目的压缩结果，本算法的平均压缩比为 3.17，平均比特数为 5.57bit，具有良好的压缩效率。

根据表 5—4 画出图 5—4，直观的表示经本文算法压缩后的曲目平均比特数。

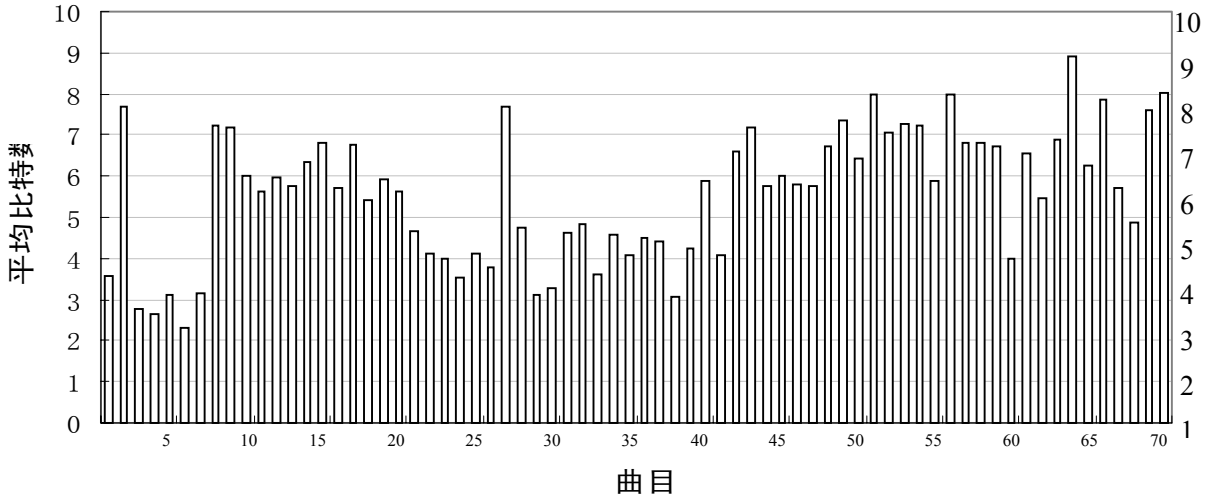


图 5—4 经本文算法压缩后的曲目平均比特数

Figure 5-4 average bit number of compressed test track

5.2.2 算法的鲁棒性

编码算法中采用分帧处理以及设计的音频帧结构，使得一旦发生数据流的损坏，损失会被限制在两帧以内，避免了误差的传播，不会造成后面所有数据的丢失。算法中对关键信息作 CRC 校验，保证了关键信息的准确性；通过实验，4 帧的编码数据出现 1 比特误码即误码率为 3.36×10^{-5} 的情况下，解压后的文件 SNR 值为 73dB，不影响聆听效果。综上所述该算法鲁棒性能良好。图 5—5 和图 5—6 分别给出不存在以及存在误码时解码得到的波形，两者只是在第一、二帧的个别样点幅值上存在差异。

5.2.3 算法的复杂度

本文提出的音频无损压缩方案，整体框架简洁，易于实现。主要的算法复杂度集中于整数 MDCT 变换部分。减少整数变换中的取整次数能增加与浮点变换结果的逼近程度，同时也能减少算法复杂度。本算法中使用的多维提升的方法，将 DCT_{IV} 的取整操作降低到 1.5N 次（N 为一帧中的样点数），加上加窗、时域混叠，IntMDCT 取整操作为 3N 次，比起经典提升的方法 DCT_{IV} 的取整操作为 $O(N\log_2 N)$次，算法复杂度大大降低。仿真结果表明，普通的 pc 机上（赛扬 2.0G 256M 内存），对于测试曲目 Track69，文件大小为 4，987，124 字节，持续时间 28.271 秒，无损音频编码时间为 20.7 秒，解码时间 18 秒。今后研究的重点就是进一步降低各个模块的算法复杂度。

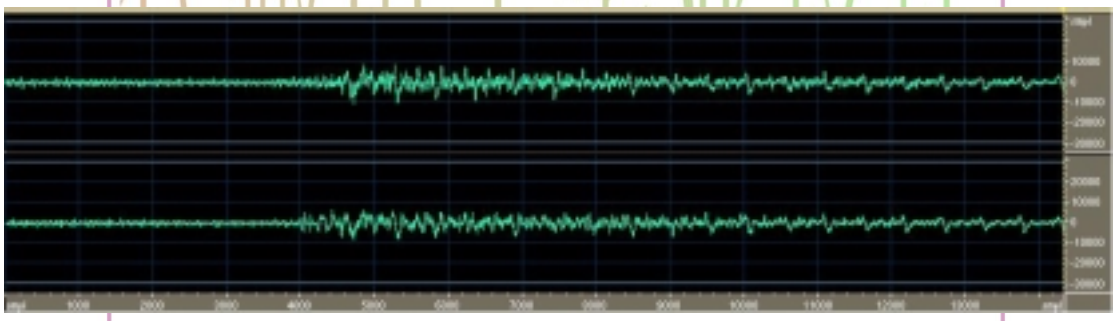


图 5－5 无误码时解码得到的波形

Figure 5-5 wave profile without bit error

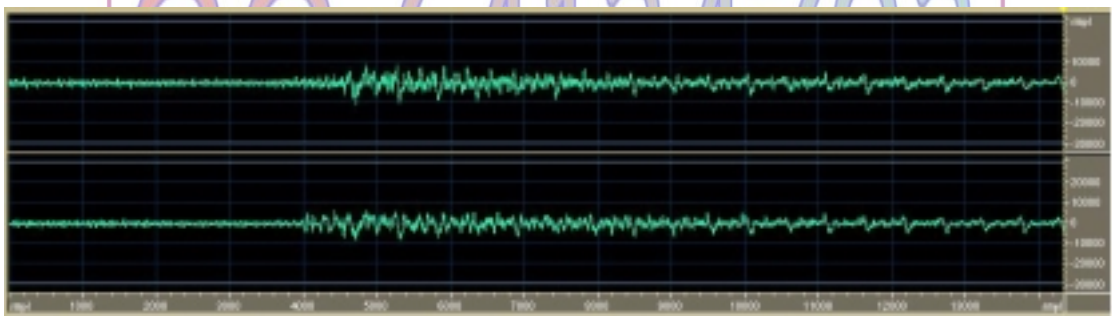


图 5－6 误码时解码得到的波形

Figure 5-6 wave profile with bit error

5.3 与现有算法压缩性能的比较

将本文提出的算法与几种现有的无损音频算法压缩效果进行比较。选用的算法是 APE（版本 3.99）、Shorten(版本 3.1)以及 LTAC（版本 1.61， 自适应块

长模式 帧长 4096)。本文提出的算法采用帧长 1024，即约 23ms 一帧进行整数变换。并且未使用联合立体声编码。实验数据为 SQAM[6]（Sound Quality Assessment Material –用于主观测试的音乐库）中多种类型的音频信号，采样率 44.1KHz，字长 16bit。

表5－5给出各种算法压缩比和平均比特数的比较。表中第1列是SQAM中各个曲目的名称，第2、3、4、5列分别列出了测试曲目经过APE、Shorten、LTAC和本文提出的算法压缩后的压缩比，第6、7、8、9列分别给出该曲目经过APE、Shorten、LTAC和本文提出的算法压缩后的平均比特数。

表 5－5 各种算法压缩比和平均比特数的比较

Table 5-5 Comparison of various algorithms – compression ratio and average bits

测试曲目	压缩比				样值平均比特数（比特）			
	APE	Shorten	LTAC	本文	APE	Shorten	LTAC	本文
Track01	10.31	2.62	2.76	4.48	1.55	6.11	5.8	3.57
Track02	2.16	1.92	2.03	2.07	7.41	8.31	7.85	7.70
Track03	16.9	7.91	7.15	5.77	0.94	2.02	2.23	2.76
Track04	11.83	7.34	5.85	6.05	1.35	2.17	2.73	2.64
Track05	6.46	1.86	2.97	5.17	2.47	8.57	5.37	3.08
Track06	16.57	8.51	6.7	6.94	0.96	1.87	2.38	2.30
Track07	6.25	4.41	4.2	5.09	2.56	3.62	3.80	3.13
Track08	2.20	2.04	2.25	2.21	7.26	7.83	7.10	7.23
Track09	2.26	2.14	2.27	2.22	7.06	7.46	7.04	7.19
Track10	2.73	2.52	2.75	2.66	5.84	6.36	5.80	6.01
Track11	3.06	2.90	2.90	2.83	5.21	5.50	5.50	5.64
Track12	2.57	2.23	2.62	2.67	6.22	7.15	6.10	5.97
Track13	2.81	2.56	2.81	2.78	5.67	6.24	5.67	5.74
Track14	2.47	2.22	2.49	2.52	6.46	7.20	6.42	6.33
Track15	2.33	2.11	2.43	2.35	6.85	7.55	6.56	6.80
Track16	2.74	2.54	2.88	2.8	5.82	6.28	5.55	5.71
Track17	2.36	2.18	2.45	2.36	6.76	7.32	6.53	6.76
Track18	3.13	2.89	3.00	2.95	5.09	5.52	5.32	5.40
Track19	2.96	2.73	2.78	2.71	5.40	5.85	5.75	5.90

Track20	2.96	2.69	2.975	2.83	5.40	5.94	5.37	5.64
Track21	3.56	3.13	3.66	3.43	4.48	5.10	4.36	4.66
Track22	4.41	4.00	4.03	3.90	3.62	3.99	3.96	4.09
Track23	4.41	4.09	3.86	4.03	3.62	3.90	4.14	3.97
Track24	5.17	4.81	4.39	4.52	3.08	3.32	3.64	3.53
Track25	4.28	4.00	4.06	3.89	3.73	3.99	3.93	4.11
Track26	4.70	4.07	4.89	4.22	3.40	3.92	3.26	3.79
Track27	2.34	1.99	2.18	2.08	6.83	8.02	7.31	7.67
Track28	3.77	3.60	3.85	3.38	4.23	4.44	4.14	4.73
Track29	6.53	7.55	7.32	5.16	2.44	2.11	2.18	3.09
Track30	6.14	5.45	6.43	4.91	2.60	2.93	2.48	3.25
Track31	3.86	3.39	4.00	3.47	4.13	4.70	3.99	4.60
Track32	3.26	2.69	3.45	3.31	4.89	5.94	4.62	4.82
Track33	5.19	4.54	5.16	4.43	3.08	3.53	3.09	3.61
Track34	4.05	3.53	3.85	3.50	3.94	4.52	4.15	4.57
Track35	4.00	2.60	3.54	3.93	4.00	6.14	4.51	4.06
Track36	3.93	3.37	4.04	3.54	4.06	4.74	3.95	4.51
Track37	3.90	3.55	3.55	3.61	4.10	4.50	4.50	4.42
Track38	6.39	6	6.75	5.22	2.50	2.66	2.36	3.06
Track39	4.12	3.74	3.98	3.76	3.88	4.27	4.01	4.25
Track40	2.70	2.54	2.89	2.71	5.92	6.28	5.53	5.90
Track41	4.16	3.79	3.92	3.92	3.84	4.22	4.08	4.08
Track42	2.35	2.16	2.88	2.42	6.78	7.30	5.54	6.58
Track43	2.15	1.97	2.31	2.22	7.43	8.08	6.91	7.2
Track44	2.92	2.55	2.82	2.78	5.46	6.26	5.68	5.75
Track45	2.79	2.42	2.68	2.66	5.73	6.59	5.95	6.02
Track46	2.85	2.37	2.71	2.76	5.60	6.72	5.90	5.79
Track47	3.05	2.62	2.84	2.78	5.28	6.10	5.62	5.76
Track48	2.45	2.24	2.42	2.39	6.51	7.13	6.59	6.70
Track49	3.14	2.12	2.19	2.18	5.08	7.53	7.29	7.33
Track50	3.47	2.47	2.56	2.49	4.60	6.47	6.24	6.42
Track51	2.61	1.96	2.25	2.49	6.12	8.13	7.10	7.97

Track52	2.99	2.24	2.26	2.26	5.34	7.11	7.06	7.06
Track53	3.12	2.17	2.21	2.20	5.12	7.34	7.23	7.28
Track54	3.19	2.20	2.23	2.21	5.01	7.26	7.15	7.21
Track55	2.87	2.54	2.73	2.71	5.56	6.28	5.86	5.89
Track56	1.99	1.68	1.99	2.00	8.03	9.5	8.03	8.00
Track57	2.33	1.79	2.41	2.35	6.84	8.915	6.61	6.78
Track58	2.65	2.37	2.38	2.35	6.01	6.73	6.71	6.79
Track59	2.40	2.14	2.39	2.37	6.65	7.44	6.68	6.72
Track60	4.59	4.07	4.31	4.02	3.48	3.92	3.70	3.97
Track61	2.52	2.21	2.45	2.43	6.35	7.25	6.50	6.56
Track62	3.14	2.69	2.94	2.93	5.07	5.94	5.42	5.46
Track63	2.49	2.16	2.31	2.32	6.40	7.39	6.90	6.88
Track64	1.81	1.66	1.80	1.80	8.80	9.61	8.84	8.91
Track65	2.73	2.44	2.68	2.55	5.85	6.54	5.95	6.26
Track66	2.12	1.91	2.03	2.03	7.52	8.33	7.86	7.86
Track67	2.90	2.67	2.80	2.79	5.50	5.97	5.70	5.72
Track68	3.59	3.29	3.35	3.28	4.45	4.85	4.77	4.87
Track69	2.18	1.95	2.10	2.11	7.32	8.16	7.59	7.59
Track70	2.25	1.93	2.00	1.99	7.08	8.26	7.97	8.02
平均	3.94	3.08	3.27	3.17	5.05	5.99	5.46	5.57

由表 5—2 可以看到，虽然比起最流行的 APE 格式的压缩率稍微有一些差距，总体上本算法对于各种音频信号均有较好的压缩效果，特别是对于乐器音频（Track08—Track43）中的部分音频曲目的压缩率甚至超过了 APE。它揭示了基于变换的无损音频压缩方案在技术上的优势。

5.4 本章小结

本章介绍了无损音频压缩算法的实现，通过实验结果的分析，选择合适的编码参数，得到最高的压缩效率。从压缩效率、计算复杂度、算法鲁棒性等方面进行本算法的性能分析。最后将本算法与现有其他算法如 APE、Shorten 和 LTAC 的压缩性能进行比较，得出结论：本算法对于各种音频类型均具有良好的压缩性能，特别是对乐器音频的压缩效率部分超过了目前最流行的 APE 格

式。

第六章　总结和展望

6.1 研究工作总结

本论文对整数变换特别是整数 MDCT 变换进行了深入研究，并将之用于无损音频压缩，取得了良好的压缩效果。总结本论文的工作，包括以下几个方面：

第一， 论文证明了 MDCT 分解为 Givens 旋转和 DCT 的推导过程，在此基础上使用两种提升方案－－“经典提升”和“多维提升”实现 IntMDCT。同时论文对 IntMDCT 中复杂的矩阵运算提出了相应简化算法，降低了算法实现的复杂度。

第二， 本文对变换后的系数进行了映射处理，并在大量统计分析的基础上，发现经过映射后的 IntMDCT 系数的概率分布接近于几何分布，因此算法中采用了 Golomb-Rice 熵编码技术，并且达到较好的编码性能。由于不同频段内变换系数的幅度差异，本论文对于不同频段的变换系数采用了不同的 Golomb-Rice 编码参数以得到最优的压缩效率。

第三， 本文提出并实现了一种新的无损音频编解码算法，设计了适合于该无损音频压缩算法的比特流结构，实现音频数据的有效编码。分析了采用两种提升实现 IntMDCT 相对于浮点变换结果的逼近误差，以及对于整个算法压缩性能的影响。

第四， 论文对本算法的压缩效率、鲁棒性和复杂度等性能进行分析，并与现有无损算法相比较。本算法对于各种音频信号均有较好的压缩效果，特别是对于乐器音频的压缩甚至超过了 APE。

6.2 进一步工作的展望

本文的工作在以下方面还有待完善：

第一， 对于整数 MDCT 可以进一步深入研究，如，研究 MDCT 的块长的选择机制，窗函数（窗的形状，窗的大小）的切换准则

第二， 对于本文提出的无损音频编码算法进一步研究，如考虑不同的熵编

码方法或者使用混合的熵编码方法，并为之确定切换准则。

第三， 对于本文提出的无损音频解码算法进一步研究，通过程序优化，进一步降低各个模块的运算复杂度，并与现有算法的解码程序的复杂度进行比较，说明其解码时间上的优势。

第四， 考虑自适应神经网络在无损压缩中的应用，必将进一步推动无损音频压缩的发展和广泛应用。

致谢

首先，衷心感谢我的导师胡剑凌副教授。本论文的全部工作都是在胡老师的精心指导下进行的。与胡老师的接触中，他的卓越的学术水平、科学的思维方式、严谨的治学态度和勤奋的工作精神给我留下了极深刻的印象，使我知道了要成为一名优秀的研究人员所应具备的素质，也为我树立了今后奋斗的榜样。在三年的研究生生活中，胡老师自始至终都给予了我无微不至的关怀，热情、耐心的指导和鼓励，使我受益匪浅，本论文的顺利完成与这些因素是分不开的。胡老师的谆谆教导和鼓励，我一直铭记在心，它们随时都激励、鞭策着我，使我不畏艰难、努力工作，这对我将来的发展是大有益处的。在此，我向胡老师再一次表示感谢！

同时感谢徐盛副教授，给予了我许多诚恳而无私的指导和帮助。感谢教研室的李力利老师和相关课题的同学，他们在工作中给予了我许多有益的指导和帮助。

感谢实验室其他的老师和同学为我创造了良好的科研氛围和愉快的生活环境。对以上的老师和同学致以真诚的谢意。

最后，衷心感谢我亲爱的父母，他们在生活和学业上给了我无私的关怀与鼓励，感谢他们在我成长道路上所付出的心血。此时此刻，我惟有希望此文能够稍微回报他们无比的关心和爱护！

攻读硕士学位期间发表的论文

1．朱敏，胡剑凌，徐盛，“基于正交变换的无损音频压缩”，电声技术，2005 年第 4 期，总第 238 期，P51-54

2．尤扬，胡剑凌，朱敏等，“基于定点 DSP 的子带分析滤波器快速算法”，电声技术，2006 年第 2 期

参考文献

[1] http://flac.sourceforge.net/

[2] A.Wegener, “MUSICompress: Lossless, low-MIPS audio compression in software and hardware,” in Proc. Int, Conf. Signal Processing Applications and Technology, 1997, Availabel:http://members,aol.com/sndspace

[3] A.Bruekers, A.Oomen, and R.van der Vleuten, “Lossless coding for DVD audio,” in Proc. 101st AES Conv.,Los Angeles, CA, Nov. 1996, prprint 4358

[4] Se Yil Park, Se Hyoung Park, Dae Sik Lim and Jaeho Shin：“A Lossless and Lossy Audio Compression using Prediction Model and Wavelet Transform” The 2002 International Technical Conference On Circuits/Systems, Computes and Communications

[5] Tilman liebchen，Marcus Purat，and Peter Noll：“Improved Lossless Transform Coding of Audio Signals” In: Impulse und Antworten - Festschrift für Manfred Krause, Wissenschaft & Technik Verlag, Berlin, 1999

[6] Sweldens. W. The lifting scheme: A construction of second generation wavelets. IMI Technical Report 1995, 06, Industrial Mathematics Initiative, Department of Mathematics, University of South Carolina, 1995

[7] INGRID DAUBECHIES AND WIM SWELDENS：“FACTORING WAVELET TRANSFORM INTO LIFING STEPS”J. Fourier Analysis and Applications, 4(3):247-269, 1998

[8] W. H. Chen, C. H. Smith, and S. C. Fralick, “A fast computational algorithm for the discrete cosine transform,” IEEE Trans. Commun, vol. COM-25, pp. 1004-1009, 1977

[9] “Reconsideration of ‘A fast computational algorithm for the discrete cosine transform.’”, IEEE Trans, Commun., vol. COM-31, pp. 121-123, 1983

[10] Z. Wang, “Fast algorithms for the discrete w transform and for the discrete fourier transform,” IEEE Trans. ASSP, vol. ASSP-32, no. 4, pp. 803-816, 1984

[11] Ralf Geiger, Yoshikazu Yokotani, Gerald Schuller, Jurgen Herre, “Improved integer transforms using mutli-dimensional lifting”, Acoustics, Speech, and Signal Processing, 2004. Proceedings. (ICASSP '04). IEEE International Conference on, Volume 2, 17-21

May 2004 Page(s):ii - 1005-8 vol.2

[12] Ralf Geiger, Jurgen Herre, Jurgen Koller, karlheinz Brandenburg：“INTMDCT－A LINK BETWEEN PERCEPTUAL AND LOSSLESS AUDIO CODING”Acoustics, Speech, and Signal Processing, 2002. Proceedings.(ICASSP ‘02). IEEE International Conference on, Volume: 2, 2002 Pages: 1813~1816

[13] Peter Fenwick：“Punctured Elias Codes for variable-length coding of the integers” Technical Report 137 ISSN 1173-3500 December 5, 1996

[14] R. Geiger, Y. Yokotani, and G. Schuller, “Improved integer transforms for lossless audio coding”, Proc. of the Asilomar Conf. on Signals, Systems, and Computers, 2003

[15] R. Geiger, T. Sporer, J. Koller, and K. Brandenburg, “Audio coding based on integer transforms,” in 111th AES Convention, New York, 2001

[16] Mat Hans and Ronald W. Schafer, “Lossless Compression of digital audio ”, IEEE SIGNAL PROCESSING MAGAZINE，JULY 2001

[17] Golomb S, “Run-length encodings”, Information Theroy, IEEE Transactions on Volume 12, Issue 3, Jul 1966 Pages:399-401

[18] Sound Quality Assessment Material – Recordings for subjective Tests. Technical Centre of the European Broading Union(EBU) , 1988

[19] Tilman liebchen，Marcus Purat，and Peter Noll：“Improved Lossless Transform Coding of Audio Signals” In: Impulse und Antworten - Festschrift für Manfred Krause, Wissenschaft & Technik Verlag, Berlin,1999

[20] ISO/IEC International Standard IS 11172-3. Information Technology-Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to About 1.5Mbit/s-Part 3:Audio, 1992

[21] Lin Xiao, Tan Wee Hong, Gee Chein Woei, Li Gang, “A Lossless Audio Compression Software for Windows Application”, Signal Processing Proceedings, 1998, ICSP’98 1998 Fourth International Conference on Volume 2, 12-16 Oct. 1998 Pages: 1162-1165 vol.2 Digital Object Identifier 10.1109 COSP. 1998. 770824

[22] Liebchen, T., Reznik, Y.A.. MPEG-4 ALS: an emerging standard for lossless audio coding. Data Compression Conference, 2004, Proceedings(DCC’04). March: 439 – 448.

[23] M. Hans and R.W. Schafer, “AudioPak—An integer arithmetic lossless audio coedec,” in

Proc. Data Compression Conf, Snowbird, UT, 1998, p.550

[24] Tony Robinson: "SHORTEN:Simple lossless and near-lossless waveform compression" Technical report CUED/F-INFENG/TR.156 ,Cambridge University Engineering Department, Trumpington Street, Cambridge, CB2 1PZ, UK, December 1994

[25] Tilman liebchen，Marcus Purat，and Peter Noll："Improved Lossless Transform Coding of Audio Signals" In: Impulse und Antworten - Festschrift für Manfred Krause, Wissenschaft & Technik Verlag, Berlin,1999

[26] ISO/IEC JTC1/SC29/WG1 (1997). JPEG_LS:emerging lossless/near-lossless compression stansard for continuous-tone images JPEG Lossless

[27] P. Craven, M.Law, and J.Stuart, "Lossless compression using IIR prediction filters," in Proc. 102nd AES Conv, Munich, Germany, 1997, preprint 4415

[28] P. Craven, M. Gerzon, "Lossless coding for audio discs," J. Audio Eng. Soc, vol. 44, no. 9, , 706-720, 1996

[29] D.Lee, Personal communications on waveform archiver, Aug. 1997

[30] C. Montgomery, Personal communication on OggSquish, Aug 1997 http://www.xiph.com

[31] R. Sprague, Private communication on Sonarc, Aug. 1997.

[32] http://www.monkeysaudio.com

[33] http://www.nue.tu-berlin.de/wer/liebchen/lpac.html

[34] A. Bruekers, A. Oomen, and R. van der Vleuten, "Lossless coding for DVD audio," in Proc. 101st AES Conv., Los Angeles, CA, Nov. 1996,preprint 4358.

[35] C. Cellier, P. Chenes, and M. Rossi, "Lossless audio data compression for real time applications," in Proc. 95th AES Conv., New York, 1993, preprint 3780.

[36] M. Hans, "Optimization of digital audio for Internet transmission," Ph.D. dissertation, School Elect. Comp. Eng., Georgia Inst. Technol., Atlanta, 1998.

[37] D. Wu, E.C.Tan. "Comparison of lossless image compression algorithms", 1999 IEEE Tencon: 718-721

[38] Howard, P.G; Vitter, J.S ,"Arithmetic coding for data compression", Proceedings of the IEEE Volume 82, Issue 6, June 1994 Page(s):857 – 865

[39] Yu R, Ko C C, Rahardja S, Lin X. "Bit- plane Golomb Coding for Sources with Laplacian Distributions". Acoustics，Speech，and Signal Processing，Proceedings. (ICASSP'03). 2003

IEEE International Conference 2003, 4：277- 280.

[40] 胡航编著，语音信号处理（修订版），哈尔滨工业大学出版社

[41] 张晓玲，图像无损压缩算法初步研究，北京工业大学硕士学位论文，2001.5

英文缩写说明

DCT	Discrete Cosine Transform 离散余弦变换
MDCT	Modified Discrete Cosine Transform 改进型离散余弦变换
IntMDCT	Integer Modified Discrete Cosine Transform 整数改进型离散余弦变换
LPC	Linear Prediction Coding 线性预测编码
CRC	Cyclic Redundancy Check 循环冗余校验
SQAM	Sound Quality Assessment Material 用于主观测试的音乐库

附录

实现 CRC 校验所采用的表格：

```
crc_table [256] =
{
    0x0000, 0x8005, 0x800f, 0x000a, 0x801b, 0x001e, 0x0014, 0x8011,
    0x8033, 0x0036, 0x003c, 0x8039, 0x0028, 0x802d, 0x8027, 0x0022,
    0x8063, 0x0066, 0x006c, 0x8069, 0x0078, 0x807d, 0x8077, 0x0072,
    0x0050, 0x8055, 0x805f, 0x005a, 0x804b, 0x004e, 0x0044, 0x8041,
    0x80c3, 0x00c6, 0x00cc, 0x80c9, 0x00d8, 0x80dd, 0x80d7, 0x00d2,
    0x00f0, 0x80f5, 0x80ff, 0x00fa, 0x80eb, 0x00ee, 0x00e4, 0x80e1,
    0x00a0, 0x80a5, 0x80af, 0x00aa, 0x80bb, 0x00be, 0x00b4, 0x80b1,
    0x8093, 0x0096, 0x009c, 0x8099, 0x0088, 0x808d, 0x8087, 0x0082,
    0x8183, 0x0186, 0x018c, 0x8189, 0x0198, 0x819d, 0x8197, 0x0192,
    0x01b0, 0x81b5, 0x81bf, 0x01ba, 0x81ab, 0x01ae, 0x01a4, 0x81a1,
    0x01e0, 0x81e5, 0x81ef, 0x01ea, 0x81fb, 0x01fe, 0x01f4, 0x81f1,
    0x81d3, 0x01d6, 0x01dc, 0x81d9, 0x01c8, 0x81cd, 0x81c7, 0x01c2,
    0x0140, 0x8145, 0x814f, 0x014a, 0x815b, 0x015e, 0x0154, 0x8151,
    0x8173, 0x0176, 0x017c, 0x8179, 0x0168, 0x816d, 0x8167, 0x0162,
    0x8123, 0x0126, 0x012c, 0x8129, 0x0138, 0x813d, 0x8137, 0x0132,
    0x0110, 0x8115, 0x811f, 0x011a, 0x810b, 0x010e, 0x0104, 0x8101,
    0x8303, 0x0306, 0x030c, 0x8309, 0x0318, 0x831d, 0x8317, 0x0312,
    0x0330, 0x8335, 0x833f, 0x033a, 0x832b, 0x032e, 0x0324, 0x8321,
    0x0360, 0x8365, 0x836f, 0x036a, 0x837b, 0x037e, 0x0374, 0x8371,
    0x8353, 0x0356, 0x035c, 0x8359, 0x0348, 0x834d, 0x8347, 0x0342,
    0x03c0, 0x83c5, 0x83cf, 0x03ca, 0x83db, 0x03de, 0x03d4, 0x83d1,
    0x83f3, 0x03f6, 0x03fc, 0x83f9, 0x03e8, 0x83ed, 0x83e7, 0x03e2,
    0x83a3, 0x03a6, 0x03ac, 0x83a9, 0x03b8, 0x83bd, 0x83b7, 0x03b2,
    0x0390, 0x8395, 0x839f, 0x039a, 0x838b, 0x038e, 0x0384, 0x8381,
    0x0280, 0x8285, 0x828f, 0x028a, 0x829b, 0x029e, 0x0294, 0x8291,
    0x82b3, 0x02b6, 0x02bc, 0x82b9, 0x02a8, 0x82ad, 0x82a7, 0x02a2,
    0x82e3, 0x02e6, 0x02ec, 0x82e9, 0x02f8, 0x82fd, 0x82f7, 0x02f2,
    0x02d0, 0x82d5, 0x82df, 0x02da, 0x82cb, 0x02ce, 0x02c4, 0x82c1,
    0x8243, 0x0246, 0x024c, 0x8249, 0x0258, 0x825d, 0x8257, 0x0252,
    0x0270, 0x8275, 0x827f, 0x027a, 0x826b, 0x026e, 0x0264, 0x8261,
    0x0220, 0x8225, 0x822f, 0x022a, 0x823b, 0x023e, 0x0234, 0x8231,
    0x8213, 0x0216, 0x021c, 0x8219, 0x0208, 0x820d, 0x8207, 0x0202
}
```