
硕士学位论文

基于 Android 的
H.264/AVC 解码器的设计与实现

H.264/AVC VIDEO DECODER DESIGN
AND IMPLEMENT BASED ON ANDROID

井洪亮

哈尔滨工业大学

2010 年 12 月

国内图书分类号：
国际图书分类号：

学校代码：10213
密级：公开

硕士学位论文

基于 Android 的
H.264/AVC 解码器的设计与实现

中国知网

硕士研究生： 井洪亮
导 师： 王鸿鹏教授
申 请 学 位： 工学硕士
学 科： 计算机科学与技术
所 在 单 位： 深圳研究生院
答 辩 日 期： 2010 年 12 月
授 予 学 位 单 位： 哈尔滨工业大学

Classified Index:

U.D.C:

Dissertation for the Master Degree

**H.264/AVC VIDEO DECODER DESIGN
AND IMPLEMENT BASED ON ANDROID**

Candidate: Jing Hongliang
Associate Supervisor: Prof. Wang Hongpeng
Academic Degree Applied for: Master of Engineering
Specialty: Computer Science and
Technology
Affiliation: Shenzhen Graduate School
Date of Defence: Dec, 2010
Degree-Conferring-Institution: Harbin Institute of Technology

摘 要

手机已经成为人们生活中必不可少的电子产品。智能手机的普及使用和 3G 时代的到来,使得流媒体技术和视频通话等成为可能。但是由于无线网络自身的传输率低、网络不稳定等特点,一般的视频标准在无线网络中的传输质量很差。H.264/AVC 作为目前最先进的视频压缩标准,在压缩效率、码流质量和容错性等方面有着良好的表现,适合于无线网络的传输应用。

为了在智能手机上实现 H.264 解码,本文在研究分析几种开源解码库之后,将基于 PC 机的 H.264 解码库移植到 Android 平台。Android 智能手机操作系统是 Google 公司最新推出的一个智能手机平台。Android 手机系统无须版权费,是一个开源的平台系统。开发人员可以在该系统上快速开发,并可以利用其他开发人员的工作成果。它有自己的内存管理控制、自己支持的标准 C 等。为了在 Android 上实现流畅的视频解码,需要对解码库进行优化移植。

本文设计并实现了基于 Android 的 H.264 视频解码器,最终目的是在 Android 上实现流畅的视频解码。本文的主要创新点体现在:

- ◆ 从开源的 FFmpeg 解码库中提取 H.264 解码部分,实现解码器从 PC 机到 Android 平台的移植。
- ◆ 基于 Android 操作系统,设计实现了 H.264 解码需要的接口,并完成解码器客户端的实现。

关键词 H.264; FFmpeg; Android 操作系统

Abstract

Mobile phones have become essential electronic products in people's lives. With the coming of 3G era and the increasing use of smartphones, streaming media technology and video calling are becoming possible. However, since the low transmission rate and unstable characteristics of the wireless network, the general video standard has poor quality in the wireless network. As the most advanced video compression standard, H.264/AVC has a good performance in the compression efficiency, stream quality and fault-tolerance. It is suitable for wireless network transmission applications.

In order to realize decoding H.264 data on the smart phone, this paper analyses several open source decoding libraries and works on porting H.264 decoder based on the PC's to the Android platform. Android smart phone operating system platform is one of Google's latest launch of mobile platform, which includes an operating system, a middleware and some critical applications. Android operating system does not need any royalty changes. It is an open source platform. Developers can quickly develop on the system and can use the results of the work by other developers. It has its own memory management control, the support of the standard C far from being perfect and so on. In order to achieve decoding H.264 video data on the Android OS, we need to optimize and port the decoding library.

This paper has designed H.264 video decoder based on the Android OS, and the ultimate goal is to achieve further smooth video decoding. The main innovations of the article are as follows:

- ◆ Cut the core code of H.264 decoder in the FFmpeg library and realizing transplantation from PC to the Android platform.
- ◆ Design the interface of H.264 decoder and implement the decoder client on Android platform.

Keywords H.264, FFmpeg, Android OS

目 录

摘 要	I
Abstract	II
第 1 章 绪论	1
1.1 课题背景	1
1.2 国内外研究现状及发展趋势	2
1.3 本文主要研究工作	3
1.4 本文的结构	4
第 2 章 Android 平台及 H.264 标准简介	5
2.1 Android 开发平台	5
2.2 Android 系统平台特性和架构	7
2.2.1 Linux 内核	8
2.2.2 Lib 库和运行环境	8
2.2.3 框架层	9
2.2.4 应用层	9
2.3 H.264 标准简介	9
2.4 H.264 结构	10
2.4.1 H.264 的档次和级别	10
2.4.2 H.264 视频格式和编码数据格式	11
2.5 H.264 技术优势	13
2.5.1 更高压缩率和图像质量	13
2.5.2 网络友好性	13
2.5.3 抗误码技术	14
2.6 本章小结	15
第 3 章 H.264 算法及解码器移植实现	16
3.1 H.264 编解码原理	16
3.2 H.264 核心算法	17
3.2.1 帧内预测和帧间预测	17
3.2.2 变换与量化	22
3.2.3 熵编码	22
3.3 H.264 解码方案研究与选择	24

3.4 解码器裁剪与优化	26
3.4.1 FFmpeg 关键数据结构	26
3.4.2 H.264 解码器裁剪优化	27
3.5 解码器编译移植	28
3.5.1 生成头文件	29
3.5.2 本地 C 实现	30
3.5.3 编译本地方法	30
3.6 本章小结	31
第 4 章 解码器客户端实现	32
4.1 开发环境搭建	32
4.2 解码器整体设计	32
4.3 解码流程	33
4.4 视频显示	35
4.4.1 YUV 到 RGB565 色彩转换	35
4.4.2 RGB 数据显示	37
4.5 解码器测试结果	38
4.6 本章小结	40
结 论	41
参考文献	42
哈尔滨工业大学硕士学位论文原创性声明	45
哈尔滨工业大学硕士学位论文使用授权书	45
致 谢	46

第 1 章 绪论

1.1 课题背景

伴随着21世纪的第一个十年的过去,全球信息产业进入又一个高速发展的时代。2009年工业和信息化部给中国移动、中国联通和中国电信发放3G牌照,标志着我国正式进入第三代移动通信时代,中国的信息产业也开始迎来一个崭新的时代。随着3G时代的到来,越来越多的3G产品开始进入人们的生活,而其中影响最广泛的无疑是3G手机时代的开启。

在手机行业中,这几年最吸引眼球的当属Google的Android手机操作系统开放平台。2007年11月5日,Android手机操作系统正式发布,2008年9月22日,第一款Android手机G1正式上市,2010年5月21日Android 2.2正式发布。目前,参加Android手机开放联盟的包括摩托罗拉、HTC、三星、LG等手机厂商,中国移动、中国联通、沃达丰电信等移动运营商和高通、德州仪器、Intel等芯片厂家,其发展势头之猛可见一斑。随着3G时代的到来,Android作为一款智能手机操作系统,必将得到越来越多的手机厂商的重视,Android手机的发展前途也是一片光明。

随着3G时代的到来,过去那种以图像和文字等静态内容为主的业务必将慢慢退出历史舞台,手机流媒体等动态内容为主的业务将成为历史的主角。手机网游、手机流媒体和电子商务被认为是3G时代手机应用的三大金矿。而最受用户期待的则是手机流媒体的应用。

手机流媒体应用的最大特点就是边缓存,边播放,节省了用户的等待时间。在多媒体技术和移动通信技术蓬勃发展的今天,音频和视频无线传输成为网络服务的核心。通过手机流媒体观看电视直播,实现视频点播和视频通话将成为最吸引用户的手机业务。

H264视频标准的特点:

H264/AVC(以下简称H.264)是ISO(国际标准化组织)/IEC(国际电工技术委员会)和ITU-T(国际电信联盟)两大国际标准化组织联手制定的新一代视频压缩标准。该标准的主要目的是为了满足不同应用对高压比运动图像日益迫切的需求而制定的^[1,2]。H.264标准具有更高的图像质量和更好的压缩率、良好的网络适应性和内在的抗丢包能力、抗误码机制等。H.264标准不仅适合于有线网络的传输,同

时也适合丢包严重、时延和抖动负责的无线信道传输。

Android 操作系统的优势:

Android OS是Google发布的基于java并运行在Linux内核上的手机操作系统。其最大的特点是开源解决方案和众多厂家的支持。Google与开放手机联盟合作开发了 Android, 这个联盟由包括中国移动、摩托罗拉、高通、宏达电和T-Mobile 在内的 30 多家技术和无线应用的领军企业组成。其强大的后台支持是Android飞速发展的保证。

从Android面世以来, Android手机和Android应用数量飞速增长。2009年Android手机出货量就达到600万部。市场研究公司IDC称, 到2013年, Android手机将成为仅次于Symbian OS的第二大手机操作系统。

1.2 国内外研究现状及发展趋势

H.264自2003年3月颁布以来, 引起了学术界和产业界的广泛关注, 目前几乎所有研究视频有关的机构和公司都在从事有关H.264标准的研究和应用开发。作为一种新的视频国际标准, H.264在编码效率、图像质量、网络友好性和抗误码方面都有了很大的进步。

目前, 对于H.264标准的研究, 成果比较显著的开源研究机构主要有以下几个: 在国内, 对H.264研究最深的当属由中国视频自由组织联合会开发的T.264。在国外, 主要有由德国HHI研究所负责开发的JM系列和法国巴黎中心学校的中心研究所的学生联合开发的X264。另外, 各大厂商都有自己对于H.264标准的研究支持。

视频业务在未来几年必将成为一个迅速发展的产业。H.264标准凭借其在视频领域标准中的优越性, 业界对其研究应用和具体实现也越来越广泛。目前主要的解决方案有以下三类^[3]。

- ◆ 全硬件实现: 能实时编解码, 支持比较高的等级和应用, 如支持高清视频(1920×1088), 但是具体实现难度比较大, 是最终的发展方向和目标。
- ◆ 全软件实现: 纯软件实现, 开发过程简单, 容易控制, 但是实时性不好, 对于较高的应用, 还无法实现实时解码。
- ◆ 软硬结合: 目前来说, 是一种比较好的解决方案。标准中比较耗时的部分用硬件实现, 提高解码效率, 其他部分用软件来实现。

随着硬件性能的不断提高和视频工作者对H.264的不断优化, H.264已经开始从理论研究发展到实际应用。在Apple公司推出Ipad之后, Apple宣布不支持

Flash, 而采用H.264标准和HTML5之后, Ipad促进了H.264应用增长160%。随后, 微软IE部门的总经理Dean Hachamovitch宣称说在IE9里将只支持用H.264标准来播放HTML5视频^[4]。在IPTV产业中, 也开始慢慢由MPEG-4标准向H.264标准转变。在智能手机领域, 随着3G时代的到来, 视频通话、视频点播、视频监控与H.264相结合的增值业务的发展前景十分看好。

Android作为目前研究最火热的智能操作系统, 对其各种应用的研究开发正处一个刚刚起步但是又飞速发展的阶段。目前, 已经有58家以上的品牌及OEM厂商推出Android平台应用, 另有33家厂商表示也将在未来推出应用Android平台产品; 其中最热门的应用是智能手机, 其次为平板电脑已经机顶盒在内的新兴产品领域^[5]。因此, 结合先进的H.264视频标准和目前火热的Android智能操作系统, 可以开发出多种增值业务, 如无线手机视频楼宇监控、病人远程视频监护、视频点播等。

1.3 本文主要研究工作

本论文的开发设计是在 Android 2.1 手机操作系统和 FFmpeg0.5 解码库提供的 H.264 解码核心算法为基础进行的。本论文的主要工作包括以下几个部分。Android OS 平台的研究、H.264 编解码原理、H.264 解码库移植与优化和视频解码器的仿真实现。

◆ Android OS 平台

Android 手机操作系统是基于 java 并运行在 Linux 内核上的。目前还不支持直接使用 C 语言的开发工作。因为, 更好的完成解码器的开发工作, 在熟悉整个 Android 架构的基础上, 重点熟悉其提供的 NDK 交叉编译工具, 完成解码器在底层使用 C 语言解码的实现。

◆ H.264 编解码原理

熟悉整个 H.264 标准的编解码过程, 重点对解码原理和算法进行详细的研究。为后期的移植优化工作提供良好的理论基础。

◆ H.264 解码库的移植

以 FFmpeg 0.5 版本为基础, 提取其中的 H.264 解码的核心算法, 进行优化移植到 android 系统中。

◆ 视频解码器的仿真实现

以移植的 H.264 解码器为基础, 结合 android SDK 提供的 API 接口和 JNI 机制, 完成解码器的仿真实现。

本论文的开发设计以 Android 2.1 版本的手机操作系统为基础进行的。本课题主要完成基于 Android 智能手机操作系统的 H.264 视频解码器的设计与实现。通过对 H.264 编解码中关键技术的研究，移植开源解码器 FFmpeg 到 Android 智能手机平台实现 H.264 视频解码的设计方案。通过学习 Android 平台的软件开发的相关知识，在 Android2.1 模拟器中编程实现视频解码器，并对其进行优化，实现流畅的视频解码。

1.4 本文的结构

本文共分 4 章：

第 1 章 绪论，介绍了课题的背景、研究现状以及本论文的主要研究内容和结构。

第 2 章 主要分两部分进行了讲解。首先详细介绍最近非常热门的 Android 智能手机平台。重点对其特性和架构进行分析，为后面解码器客户端的实现提供良好的基础。其次，对 H.264 标准进行了概述，对其特性和技术优势进行了阐述。

第 3 章 对 H.264 编解码原理进行了阐述，然后对编解码中的重点算法和环节进行详细的理论研究和分析，在理解 H.264 算法的基础之上，对几种开源的 H.264 解码器进行了比较和分析，确定移植方案，基于 FFmpeg 的 H.264 解码器为原型，进行裁剪优化，最后给出了解码器的移植过程。

第 4 章 从搭建 Android 的开发环境入手，利用前面移植的 H.264 解码器为核心，结合 Android 的 java 层提供的组件，在 Android 2.1 模拟器上，仿真实现了 H.264 视频解码播放功能。

第 2 章 Android 平台及 H.264 标准简介

2.1 Android 开发平台

移动通信技术至今已有 20 多年的历史，从第一代模拟移动通信到第二代数字移动通信，再到目前如火如荼的 3G 技术，全球迎来了移动通信发展的高峰。无线宽带接入技术、Wi-Fi 等，这些技术的不断发展为无线高速上网、流媒体技术、无线视频传输这些大数据业务的实现奠定了基础^[6,7]。随着 3G 时代的到来，智能手机市场也进入了一个高速发展，百家争鸣的时代。

所谓的智能手机，是指使用开放式操作系统的手机，同时第三方可根据操作系统提供的编程接口为手机开发各种扩展应用和提供各种扩展硬件^[8]。目前市场上主流的智能机操作系统有 Symbian、Linux、Windows Mobile、Palm、iPhone 和 BlackBerry 等。Android 智能手机操作系统是最近几年刚刚出现并迅速流行的一种智能手机操作系统。

1. Symbian

Symbian 是由诺基亚、西门子等几家大型移动设备通信商共同出资组建的一个合资公司，专门研发手机操作系统^[9]。毫无疑问，Symbian 系统是现今手机中应用最为广泛的操作系统。尤其是 Nokia 手机的流行，极大的推动了 Symbian 系统的广泛应用。

2. Linux

Linux 具有源代码开放、软件授权费用低、应用开发人才资源丰富等优点，再加上 IBM、Sun 等计算机巨头的支持，使得 Linux 在手机操作系统中异军突起^[9]。但是由于其介入手机领域比较晚，因此发展规模不是很大，目前只有少数机型使用的是 Linux 手机操作系统。Android 手机操作系统就是基于 Linux2.6 内核的。

3. Windows Mobile

Windows Mobile 前身就是微软 1996 年推出的 Windows CE，到了 2003 年又开发出 Pocket Phone Edition 和 PowerSmartPhone,三者融合才产生了严格意义上的 Windows Mobile. Windows Mobile 由于其背后依托 windows，因此具有强大的先天优势，譬如在内建软件方面，但是其对手机硬件要求高，在能耗方面要明显逊于其他操作系统。

4. Palm

Palm 操作系统是 Palm 公司开发的专用于 PDA 上的一种 32 位的嵌入式操作系统，它的操作界面采用触控式，差不多所有的控制选项 都排列在屏幕上，使用触控笔便可进行所有操作。Palm 最大的优点是界面简单、系统反应速度快。但是其本身并不具备录音、MP3 等功能。

5. iPhone

iPhone OS 是由苹果公司专门为 iPhone 开发的操作系统，主要应用于 iPhone 和 iPod touch。iPhone 操作系统的系统架构主要分为四个层次。系统操作占用大概 512MB 的存储空间。 iPhone OS 是一款具有革命性的、划时代的操作系统， iPhone 给手机用户的体验达到了前所未有的境界。

6. BlackBerry

黑莓 BlackBerry 是美国市场占有率第一的智能手机，它的经典设计就是宽大的屏幕和便于输入的 QWERTY 键盘，Blackberry 与桌面 PC 同步堪称完美，它可以自动把你 Outlook 邮件转寄到 Blackberry 中，所以 BlackBerry 一直是移动电邮的巨无霸。正因为是正统的商务机，所以它在多媒体播放方面的功能非常孱弱，也许它在未来应该着力改善这个弱点，因为手机功能的整合是大势所趋，人们不会只满足于单一的功能。

Android 一词的本意是指“机器人”，同时也是 Google 开发的基于 Linux 平台的开源手机操作系统^[10]。该平台由操作系统、中间件、用户界面和应用程序组成。Android 是完全开源的，不存在任何以往阻碍移动产业创新的专有权障碍。

Android 将手机的普及性、开源软件的活跃性，以及谷歌和其他开放手机联盟成员(如英特尔、TI、T-Mobile 和 NNT DoCoMo)的集体支持完美的结合在一起^[11]。这让全世界的电信业者、手机硬件制造商及其网络服务业掀起了“给我 Android，其余免谈”的热潮，也让全世界的程序员团结了起来，Android 是一个你不可不学的移动平台。

总得来说，Symbian 太强势、Windows Mobile 太贵、iPhone 太封闭都是相比 Android 之下的弱点。因此相对而言，Android 的免费对手机厂商吸引力还是很大的，而且运营商也希望通过开源免费的 Android 定制自己的操作系统以达到控制终端业务的目的。易观智库 Enfodesk 近期发布数据显示，中国市场 Android 平台发展迅速，2010 年第三季度保有量达到 861.2 万部，Symbian 则出现下滑态势，第三季度约占整个智能手机市场的 23.5%，较上一季度环比下降 3.1 个百分点^[12]。

2.2 Android 系统平台特性和架构

应用程序框架：支持组件的重用和替换。

Dalvik 虚拟机：Dalvik 虚拟机是 Google 自己开发的用于 Android 平台的 JAVA 虚拟机。不同于 Java 虚拟机运行 java 字节码，Dalvik 虚拟机运行的是其专有的文件格式 Dex。Dalvik 是基于寄存器的经过优化的虚拟机，允许在有限的内存中同时运行多个虚拟机的实例，并且每一个 Dalvik 应用作为一个独立的 Linux 进程执行。独立的进程可以防止在虚拟机崩溃的时候所有程序都被关闭^[13]。

集成的浏览器：基于开源的 WebKit 引擎。

优化的图形库：自定义的 2D 图形库，3D 图形库基于 OpenGL ES 1.0 (硬件可加速)。

多媒体支持：支持各种常见的音频、视频和图片格式。(MP3、AAC、MP4、3GP、JPG、PNG 等)。

SQLite：用作结构化的数据存储。SQLite 是一款轻型的数据库，是遵守 ACID 的关联式数据库管理系统，它的设计目标是嵌入式的，占用资源非常的低，在嵌入式设备中，只需要几百 K 的内存就可以。同时能够跟许多程序相结合和支持事务处理功能等。

蓝牙、Wi-Fi 技术：蓝牙是一种支持设备短距离通信（一般 10m 内）的无线电技术。能在包括移动电话、PDA、无线耳机、笔记本电脑、相关外设等众多设备之间进行无线信息交换。利用“蓝牙”技术，能够有效地简化移动通信终端设备之间的通信，也能够成功地简化设备与因特网 Internet 之间的通信，从而数据传输变得更加迅速高效，为无线通信拓宽道路^[14]。Wi-Fi 是一种可以将个人电脑、手持设备（如 PDA、手机）等终端以无线方式互相连接的技术。其目前使用的标准有两个：IEEE802.11a 和 IEEE802.11b。IEEE802.11a 定义了一个在 5GHz ISM 频段上的数据传输速率可达 54Mbit/s 的物理层，802.11b 定义了一个在 2.4GHz 的 ISM 频段上但数据传输速率高达 11Mbit/s 的物理层。

GPS、指南针和加速度计：(依赖于硬件)。

丰富的开发环境：包括调试工具、设备模拟器、内存及性能分析表，和 Eclipse 集成开发环境插件。

Android在Linux2.6内核的基础上，提供了各种函数库和一个完整的应用程序框架。在Linux核心的基础上，提供Google自己的应用程序运行环境Dalvik虚拟机。我们可以把Android大体上分为4层：Linux内核，lib库和运行环境，

框架层以及应用层。其系统架构如下图2-1表示：

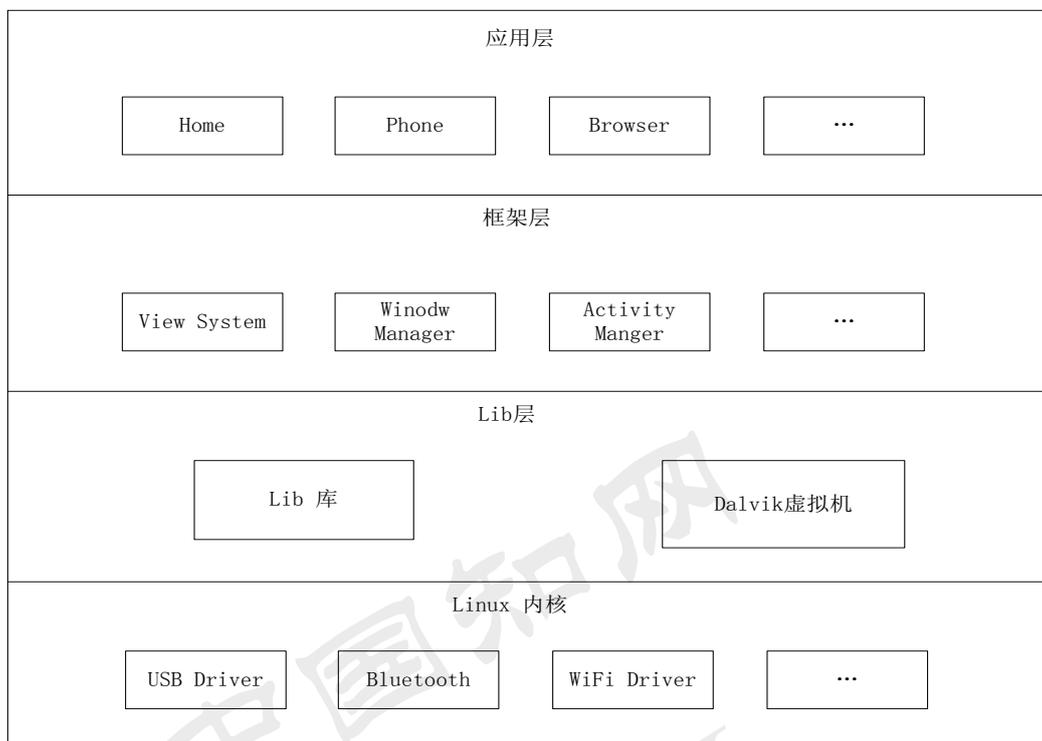


图2-1 Android系统架构

2.2.1 Linux 内核

Android依赖Linux2.6内核提供核心服务，比如内存管理、进程管理、安全机制、网络、硬件驱动等。内核扮演的是硬件层和软件层之间的抽象层的概念。

2.2.2 Lib 库和运行环境

Android提供了丰富的类库支持，被用于Android的各种组件中。这些类库主要包括：

标准C库：实现了BSD-derived的标准C库，用于底层嵌入式Linux设备驱动开发。

媒体库：基于OpenCore，该库提供了对许多常见的音频和视频的播放和录制功能，还包括对静态图片文件的处理。

WebKit：Web浏览引擎，为Android浏览器和内嵌的Web View提供实现。

2D、3D图形库：基于OpenGL ES 1.0实现的库，该库提供3D图形加速等。

SQLite：一个强大的、轻量型的关系数据库引擎。

Android运行时(Runtime)包含一组核心库，提供java运行时的运行环境。Android提供自己的Dalvik虚拟机。该虚拟机是基于寄存器的，采用简练、高

效的byte code格式运行,能够在资耗和没有应用相互干扰的情况下并行执行多个应用。每一个Android应用运行在自己的进程里面,应用自己的Dalvik虚拟机。Dalvik可以让一个设备高效的运行多个虚拟机。

2.2.3 框架层

开发者可以访问核心应用程序所使用的API框架。该应用程序框架简化了各组件之间的重用:任何一个应用程序都可以发布它的功能模块并且任何其它的应用程序都可以使用其所发布的功能块(应用的使用方法需要遵循框架的安全性限制)^[15]。同样,该应用程序重用机制也使用户可以方便的替换程序组件。隐藏在每个应用后面的是一系列的服务和系统,其中包括:丰富而且扩展性强的视图(Views)、内容提供者(Content Providers)、资源管理器(Resource Manager)、通知管理器(Notification Manager)和活动管理器(Activity Manager)等。

2.2.4 应用层

Android带有一系列用java语言编写的应用程序,包括email客户端, SMS短消息程序,地图,日历,浏览器联系人管理程序等。

2.3 H.264 标准简介

H.264/AVC 视频编解码是两大国际标准组织 ITU-T 视频编码专家组 VCEG(Video Coding Experts Group)和 ISO/IEC 运动图像专家组 MPEG(Moving Picture Experts Group)共同制定的视频编码标准^[16]。在 2001 年, VCEG 和 MPEG 联合成立了一个组织—联合视频组 JVT(Joint Video Team),这个组织的主要任务就是制定新一代的视频压缩国际标准。2003 年 3 月公布了这一标准,国际电信联盟把它叫做 H.264,运动图像专家组把它称作 MPEG4 的第 10 个部分(Part of 10 MPEG4),流传较为广泛的官方名称为 AVC(Advanced Video Coding)^[17]。

H.264 标准采用跟以往标准相似的基于块的混合编码的思想,但是在具体实现上有所区别,可以获得比 H.263++和 MPEG4 更好的压缩性;采用“网络友好”的语法和结构,增强了对各种不同网络的适应性,有利于对误码和丢包的处理;应用目标范围比较广,可以满足不同速率、不同传输以及不同存储场合的需求^[18]。H.264 也是采用变换和预测的混合编码模式。但是, H.264 标准提出了一个自己的新的概念。H.264 的功能分为两层:既视频编码层 VCL(Video

Coding Layer)和网络提取层 NAL(Network Abstraction Layer)^[19]。VCL 既编码处理的输出,表示被压缩编码后的视频数据序列,是视频内容的核心压缩。被压缩编码的 VCL 数据被封装进 NAL 单元, NAL 单元是通过特定类型网络进行传送的表述, NAL 结构便于进行信息的封装和对信息进行更好的优先级控制。

2.4 H.264 结构

2.4.1 H.264 的档次和级别

H.264是一个总的视频压缩标准,为了适应不同场合的不同应用, H.264规定了不同的档次。其每一个档次规定了不同的语法元素和句法,适合于不同的应用场合。

- ◆ 基本档次: 利用I片P片支持帧内预测和帧间预测编码,支持利用基于上下文的自适应的变长编码进行熵编码(CAVLC)。主要用于会议电视、可视电话、无线通信等实时视频通信^[20]。
- ◆ 主要档次: 支持隔行视频,采用采用加权预测的帧内编码和B片的帧间编码;支持利用基于上下文的自适应的算术编码(CABAC)。主要用于数字广播电视与数字视频存储等。
- ◆ 扩展档次: 支持码流之间的切换(SP片和SI片),改进误码性能(数据分割)、但是不支持隔行视频和自适应算术编码(CABAC)。
- ◆ 高级档次: 2004年,视频联合小组又增加了一个高端档次用于支持高精度拓展FRExt(Fidelity Range Extensions),该拓展支持更高的像素精度。

H.264的4个档次具有不同的功能,每个档次设定不同的参数(如采样速率、编码比特率、图像尺寸等),得到编解码器不同性能的级。四个档次之间的关系可以用图2-2表示。

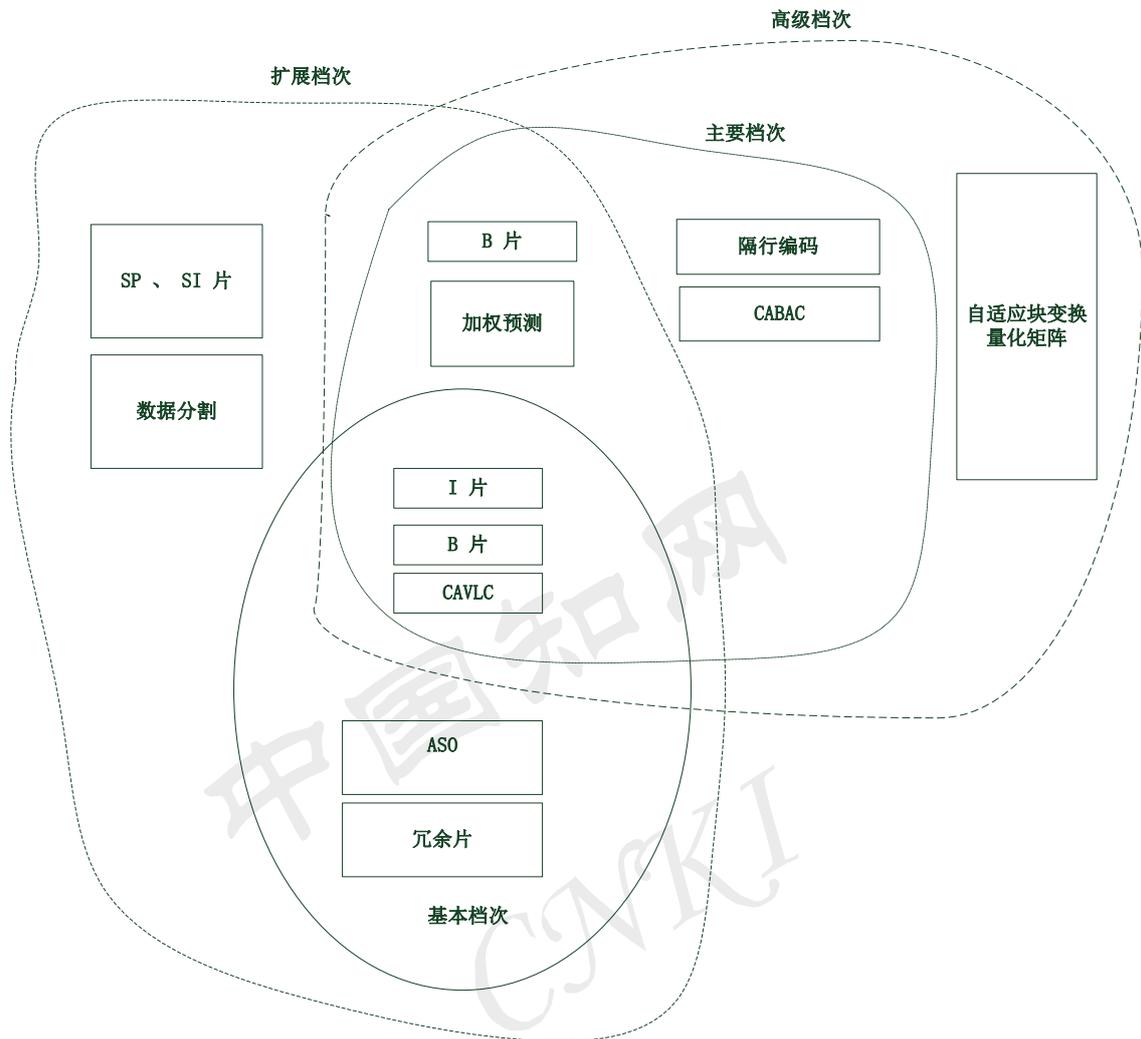


图2-2 H.264档次图

2.4.2 H.264 视频格式和编码数据格式

H.264支持编码和解码4:2:0的逐行和隔行采样的视频格式。图2-3显示的是4:2:0的逐行采样的采样格式，既水平方向和垂直方向上Cb、Cr的分辨率都只有亮度Y的一半。而图2-4显示的是隔行采样的采样格式，既顶场和底场两场中的采样点之和与逐行采样中一帧的采样点数相同^[21]。

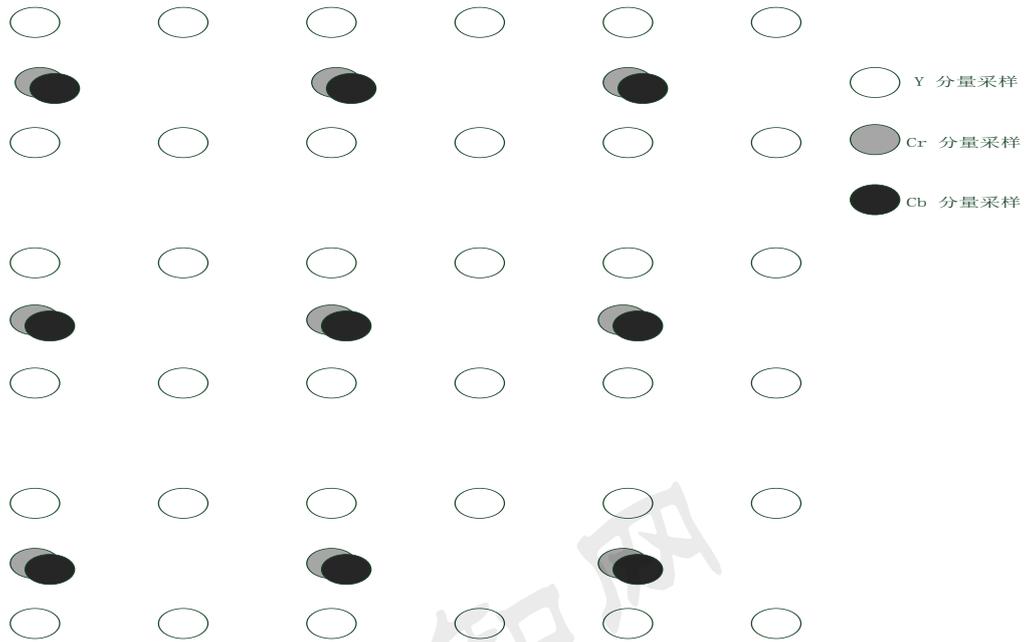


图 2-3 4:2:0 连续采样

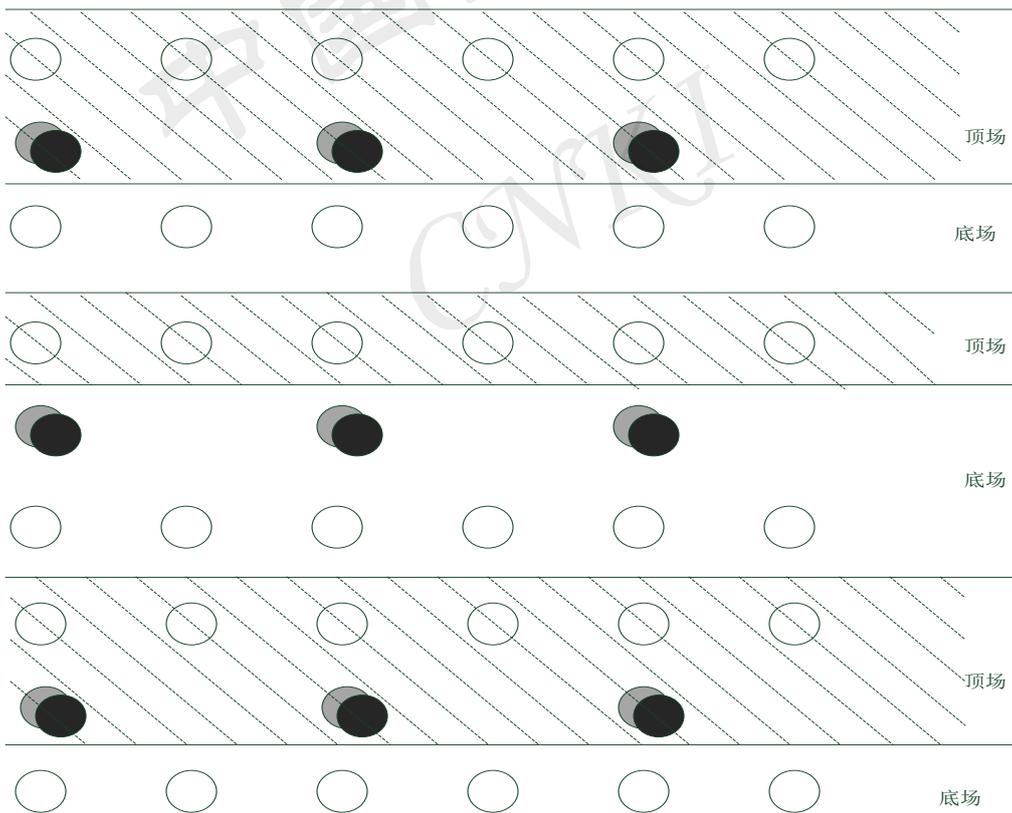


图 2-4 顶场和底场中 4:2:0 采样

H.264 没有明确规定一个编解码器是如何实现的，而是规定了编码后的视频流的句法结构和该视频比特流的解码方法，因此 H.264 标准在实现上具有很

大的灵活性。制定 H.264 标准的主要目的有以下两个：

- ◆ 在获得高图像质量的前提下进一步提高压缩比：当初联合视频组 JVT 的指标是比 H.263 和 MPEG-4 等标准具有更高的压缩比，H.264 压缩比相比他们而言压缩比提高了 2 倍。
- ◆ 更好的网络亲和性，可适用于各种不同网络传输。

为了实现“网络友好”的亲和性，H.264 的功能分为两个层次，既视频编码层 VCL(Video Coding Layer)和网络提取层 NAL(Network Abstraction Layer)。VCL 数据既编码处理的输出，它表示被压缩后的视频数据序列。VCL 层输出的是数据比特串 SODB(String Of Data Bits),SODB 字节对齐处理后封装成原始字节序列载荷 RBSP(Raw Byte Sequence Payload)。在 VCL 存贮或者传输之前，这些编码的 VCL 数据被映射或者封装在 NAL 单元中。每个 NAL 单元包括一个原始字节序列载荷 RBSP(Raw Byte Sequence Payload)，一组对应于视频编码数据的 NAL 单元头信息。NAL 单元序列的结构如图 2-5 所示。



图 2-5 NAL 单元序列结构图

2.5 H.264 技术优势

H.264标准的编码思想与现有的视频编解码标准一致：都是采用基于块的混合编码方法。但是它在借鉴MPEG系列和H系列视频标准的基础上，运用了大量不同的技术，使得其编码性能远远优于其他标准。其技术优越性主要表现在三个方面。

2.5.1 更高压缩率和图像质量

H.264标准通过对帧内预测、帧间预测、变换编码和熵编码等算法的改进和优化，在保证图像质量的基础上，进一步提高了压缩率。H.264帧间预测具有3个新的特点：1/4像素精度的运动估计，7种大小不同的可变块模式和前向与后向多参考帧模式。

2.5.2 网络友好性

为了更好的适应不同系统的应用，H.264采用了分层结构(如图2-6所示)的设计编码方式：既视频编码层VCL和网络抽取层NAL。其中，VCL用于视频编解码，包括运动补偿、变换编码和熵编码等，可以在尽可能独立于网络的情况

下获得高效的编解码。而NAL层采用统一格式对VCL数据进行封装打包，适用于不同网络中的视频传输，网络亲和性好，能很好地适应IP和无线网络的应用[22]。

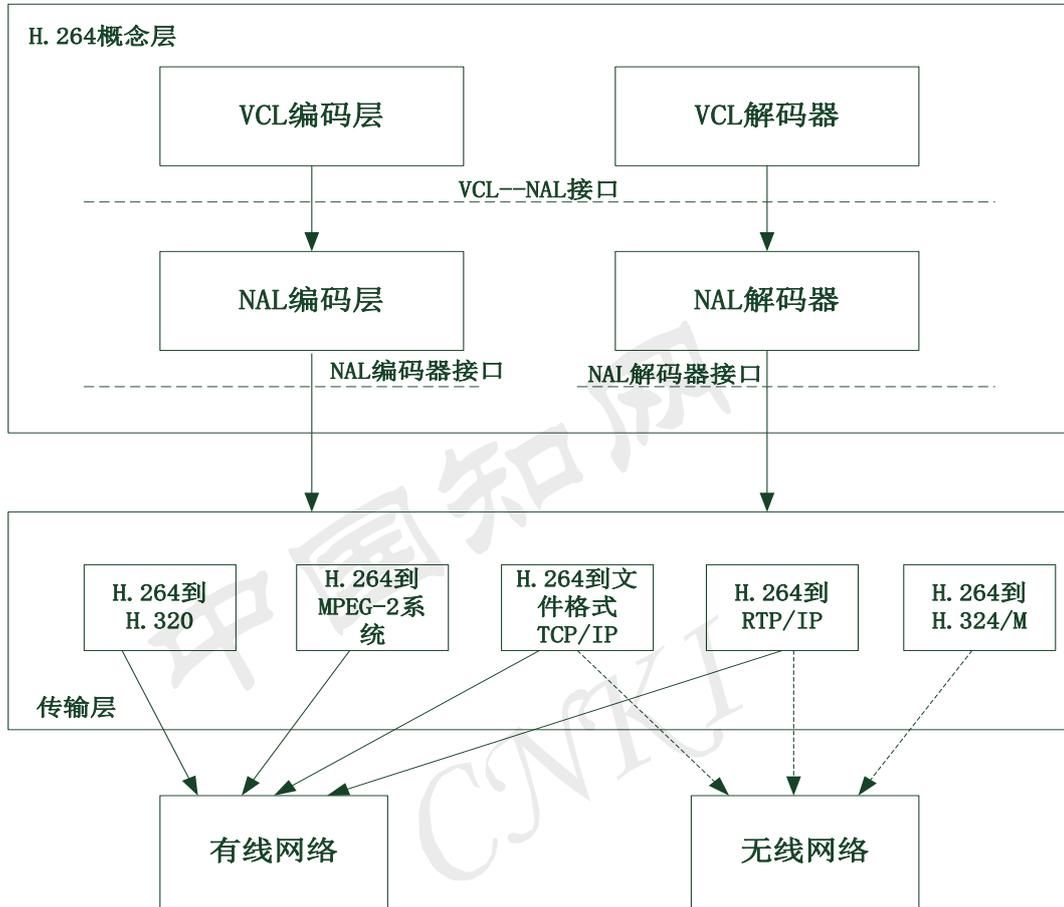


图 2-6 H.264系统分层结构图

网络抽取层(NAL)的设计目标是根据不同的网络将数据打包成相应的格式，将视频编码层(VCL)产生的比特字符串适配到各种各样的网络和多元环境中去。NAL以NALU(NAL Unit)为单元来支持编码数据在网络中的传输。H.264定义了两种封装格式。为了适应类似MPEG-2和H.320等实时通信环境，H.264定义了一种比特流格式的传输机制。每一个NAL单元由开始前缀码标识，并通过竞争控制机制来保证开始码前缀的唯一性。为了适应Internet协议RTP系统网络环境，它直接利用分组网络中的传输控制机制将NAL单元封装在包结构中，不再需要特殊的标志位来区分不同的NAL单元，从而节省了分组负载，提高了传输效率。

2.5.3 抗误码技术

在H.264中主要从3个方面来提高抗误码技术。一是直接采用其他标准中

比较成熟的抗误码技术，如图像分割、参考图像选择、帧内宏块刷新策略等。二是经过改进在 H.264 中得到更好应用的抗误码技术，如帧内编码、数据分割、分层编码等。三是 H.264 全新的抗误码技术，如片的使用、参数集、灵活宏块排序(FMO)、冗余片等关键技术可以大大提高系统的抗丢包和抗误码性能^[23]。

- ◆ 参数集：在以前的视频编解码标准中，图像头信息是至关重要的，如果包含这些信息的包丢失将直接导致与这些信息相关的数据不可用，为了防止信息头数据的丢失，以前的视频标准一般都采用了冗余编码技术来保护这些信息头。为解决这些问题，H.264 将这些很少变化并且对大量 NALU 起作用的信息放在参数集中传送。H.264 的参数集分为两种：即序列参数集(sequence parameter set)和图像参数集(picture parameter set)，序列参数集(SPS)对一系列连续编码图像起作用，图像参数集(PPS)对连续编码图像序列中的单独图像起作用。NALU 通过标识位来指定它所参考的参数集。为适应各种不同的网络环境，参数集可以采用带内传送，也可以采用带外方式传送。参数集使得编码器可以利用条带头信息本身传递重要的、不经常改变的序列参数和图像参数。参数集及其灵活的传送方式的使用大大降低了因关键的头信息丢失而造成误码的可能性。
- ◆ 冗余片：为了提高 H.264 解码器在数据丢失时的健壮性，可以采取传送冗余片的方式。这些冗余片可以是基本图像的一部分，冗余片和基本片可以采用不同的编码参数，当基本片丢失后，可以利用冗余片重构来复原图像。
- ◆ 灵活的宏块排序(FMO)：通过灵活的宏块排序 FMO 可以进一步提高差错恢复能力。通过片组的使用，FMO 改变了图像划分为片和块的方式。利用 FMO 技术，H.264 定义了 7 种宏块扫描模式。

2.6 本章小结

本章首先简要介绍了一些目前主流的智能手机系统，并对其优缺点进行了分析阐述。重点对 Google 最新推出的 Android 智能手机平台进行了介绍。对其特性和架构等方面入手，进行阐述。Android 作为目前最热门的一款智能手机平台，其发展前景非常好。然后对 H.264 标准进行了分析，因为 H.264 采用了不同于其他标准的新技术，才使得 H.264 标准具有很好的压缩率和图像质量。只有很好的了解 H.264 编码原理的基础之上，才能完成 H.264 解码器的优化移植工作。

第 3 章 H.264 算法及解码器移植实现

3.1 H.264 编解码原理

H.264 并不是明确的规定一个编解码器是如何实现的，而是规定了构成编码的比特流的语法、语法元素的语义以及语义元素的解码过程，为不同制造商的编解码器提供兼容性，各个厂商的编码器和解码器在此框架下应能互通，在实现上具有较大的灵活性，而且有利用相互竞争^[24,25]。H.264 编解码的功能模块跟一般的编解码器大致相同，主要包括预测、变换、量化和熵编码等功能模块，H.264 编解码的重要变化主要体现在各个模块的细节上，其编码原理如图 3-1 所示，解码就是编码的逆过程，其原理如图 3-2 所示。

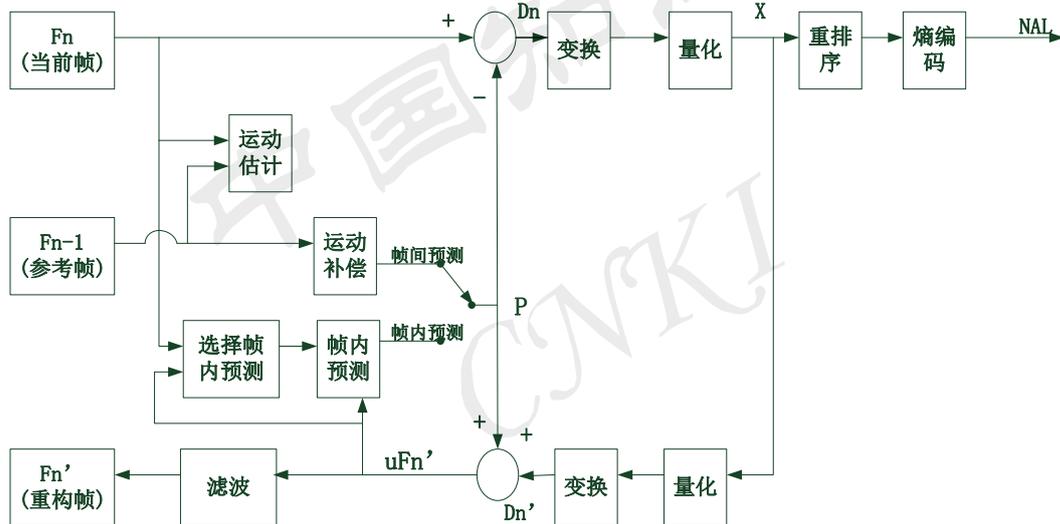


图 3-1 H.264 编码原理图

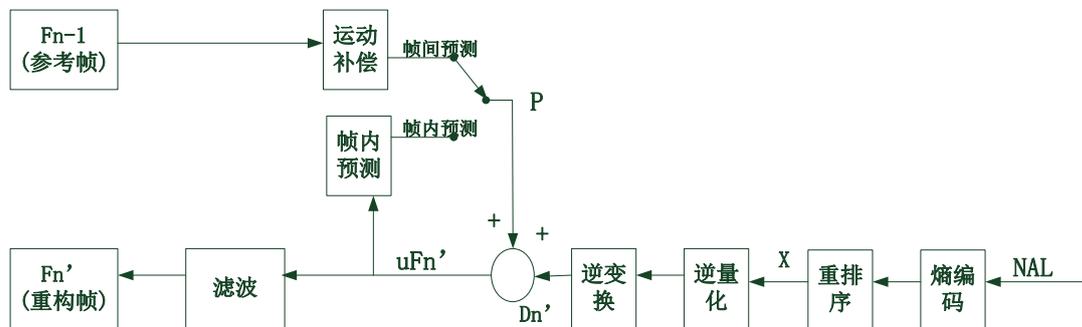


图 3-2 H.264 解码原理图

3.2 H.264 核心算法

H.264 标准的核心思想与现有的其它视频编解码标准一致,也是采用变换和预测的混合编码方法。但是, H.264 在算法的实现细节上使用了不同于其他标准的新技术,使得 H.264 编码性能远远优于其他标准。H.264 的核心算法主要包括帧内预测模式、整数变换编码、先进的量化、熵编码和高级运动估计与补偿等。对其主要的核心算法进行详细的技术分析。

3.2.1 帧内预测和帧间预测

3.2.1.1 帧内预测

在视频编码中,通常是将一幅图像划分为若干宏块进行编码。对于一幅图像而言,其像素在空间上存在很大的冗余度,消除这种空间冗余度可以很好的提高压缩率。以往的视频标准中,通常直接对宏块进行 DCT 变换,对变换系数进行熵编码^[26]。这样,就在一定程度上消除了空间冗余度,提高了压缩率。但是,这只是考虑了宏块内部像素之间的相关性,而没有考虑宏块之间的像素相关性。H.264 引入了帧内预测的方法,很好的利用了宏块之间的相关性,以进一步消除空间冗余度。

在帧内预测模式下,依据先前编码和重建之后的块形成一个预测块 P,当前块减去这个预测块,将差值进行编码。H.264 的帧内预测支持对亮度分量和色度分量独立进行编解码。对于 16×16 大小的亮度块,可以有 4 种预测模式;对于 4×4 大小的亮度块,可以支持多达 9 种的预测模式。而对于色度分量而言,支持 8×8 的块大小进行预测,有 4 种预测模式,与 16×16 的亮度块的预测模式基本相同。另外, H.264 还有一种帧内编码格式,即 I_PCM,在这种模式下编码器能够直接传输图像像素值(没有经过预测和变换)。在某些特殊情况下,这种 I_PCM 可能比通常的帧间预测、变换、量化和熵编码过程更加有效。

1. 4×4 亮度块的预测模式

图 3-3 显示了需要进行预测 4×4 亮度块的 9 种预测模式。上边和左边的像素(在图中用 A~M 标记)表示已经被编码和重建的像素,在编码器和解码器中都可以被作为预测参考。因此可以利用 A~M 的值和 9 种预测模式实现对阴影部分像素值的预测。

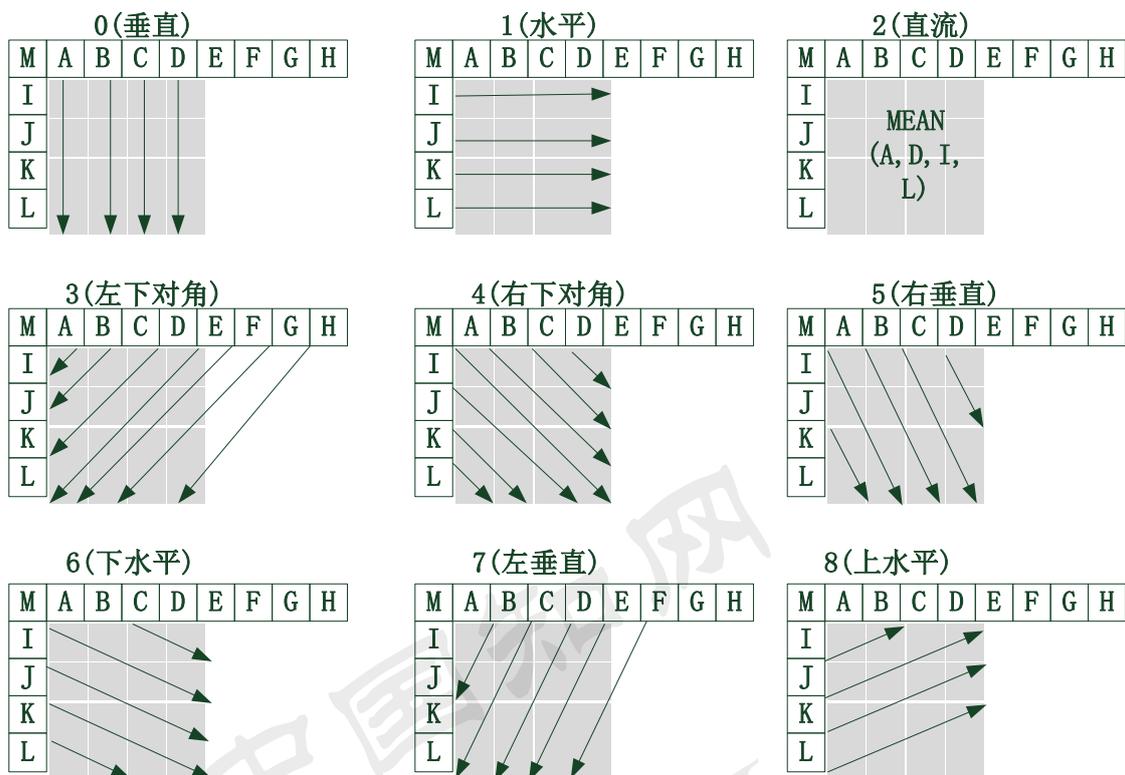


图 3-3 4X4 亮度块预测模式

表 3-1 4X4 亮度块预测模式

模式 0(垂直)	垂直推断是上样点 A,B,C,D
模式 1(水平)	水平推断是左样点 I,J,K,L
模式 2(直流)	用样点 AD 和 IL 来预测阴影内的所有样点
模式 3(左下对角)	在左下与右上之间 45° 角度, 插补样点
模式 4(右下对角)	在又 45° 角下推断样点
模式 5(右垂直)	在垂直向左近 26.6° (宽/高=1/2)推断样点
模式 6(下水平)	在水平下近 26.6° 推断样点
模式 7(左垂直)	在垂直向右近 26.6° (宽/高=2/1)推断样点
模式 8(上水平)	在水平上近 26.6° 推断样点

2. 16×16 亮度块的预测模式

同样, 一个宏块的全部 16×16 亮度分量可以通过其左边和上面的 32 个像素的重构值进行预测。其预测模式如图 3-4 所示。

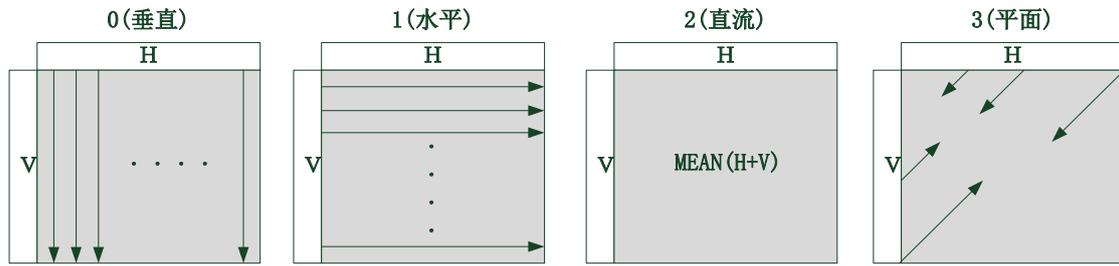


图 3-4 16X16 亮度块预测模式

表 3-2 16X16 亮度块预测模式

模式 0(垂直)	从上面的像素推断(H)
模式 1(水平)	从左边的像素推断(V)
模式 2(直流)	上面和左边的像素的平均值
模式 3(左下对角)	对上面 H 和左边 V 像素使用一个线性平面函数，这在亮度的平滑变化区域效果好

3.2.1.2 帧间预测

帧内预测是从空间上消除冗余性，而帧间预测则是从时域考虑，消除冗余性达到提高压缩率的目的。帧间预测是采用基于块的运动补偿从一个或者多个先前编码的图像帧中产生一个预测模型。H.264 标准与其它标准不同，它支持不同块大小的树形结构的运动补偿和 1/4 像素精度的运动估计等新技术，使预测精度得到了极大的提高。

1. 树形结构的运动补偿

H.264 在运动估计时，可以灵活的选择宏块的大小。每个 16×16 的宏块可以有 4 中划分方式：一个 16×16，两个 16×8，两个 8×16 或者四个 8×8。当划分为 8×8 时，又可以进一步采用两个 8×4，两个 4×8 和四个 4×4 三种块分割^[27]。H.264 的这种分割叫做树状结构运动补偿(分割方式如图 3-5 所示)。这样做既可以使运动物体的划分更加精确，减小运动物体边缘的衔接误差，又可以减小变换过程中的计算量。实验结果表明，与仅仅采用 16×16 的块相比，采用这种可变块大小的树状结构运动补偿模式可以节约 16%的码率^[28]。

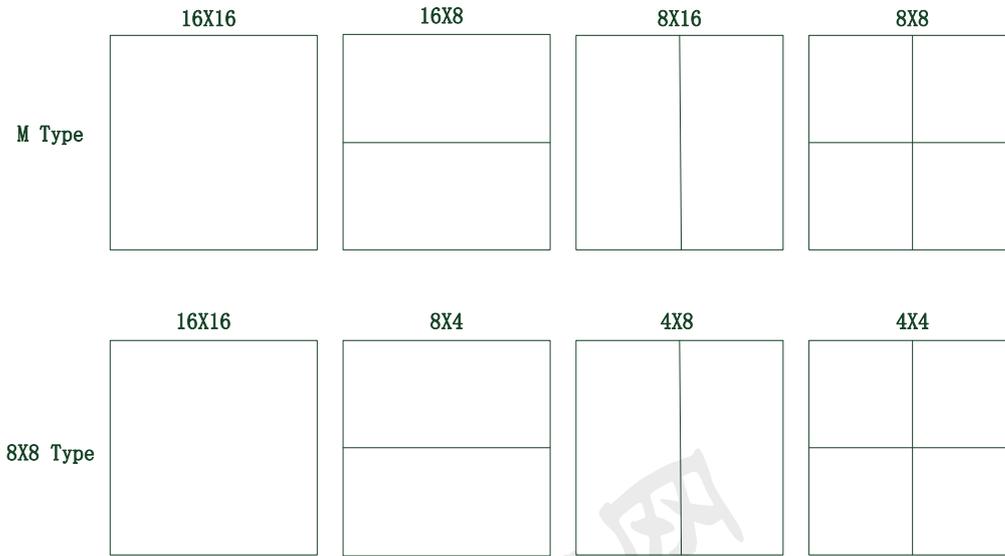


图 3-5 宏块及子宏块分割图

2. 1/4 像素精度的运动估计

由于图像在时间上的连续性,使得相邻的两幅图像之间块的运动矢量具有一定的连续性。H.264 标准相对于其他标准而言,支持更为精确的 1/4 像素精度(色度块采用 1/8 像素精度)的运动矢量。H.264 的 1/4 像素精度是通过线性插补得到的。首先产生整数像素之间的 1/2 像素点(图 3-6 中灰色表示)。1/2 像素是通过利用一个六阶有限冲击响应滤波器对相邻整数位置的像素值进行内插得到的,其权重值是 $(1/32, -5/32, 5/8, 5/8, -5/32, 1/32)$ ^[29]。对于图 3-11 中的半像素点 b,其差值公式表示如下式 3-1(公式中 round 表示舍入):

$$b = \text{round}\left(\frac{E - 5F + 20G + 20H - 5I + J}{2}\right) \quad (3-1)$$

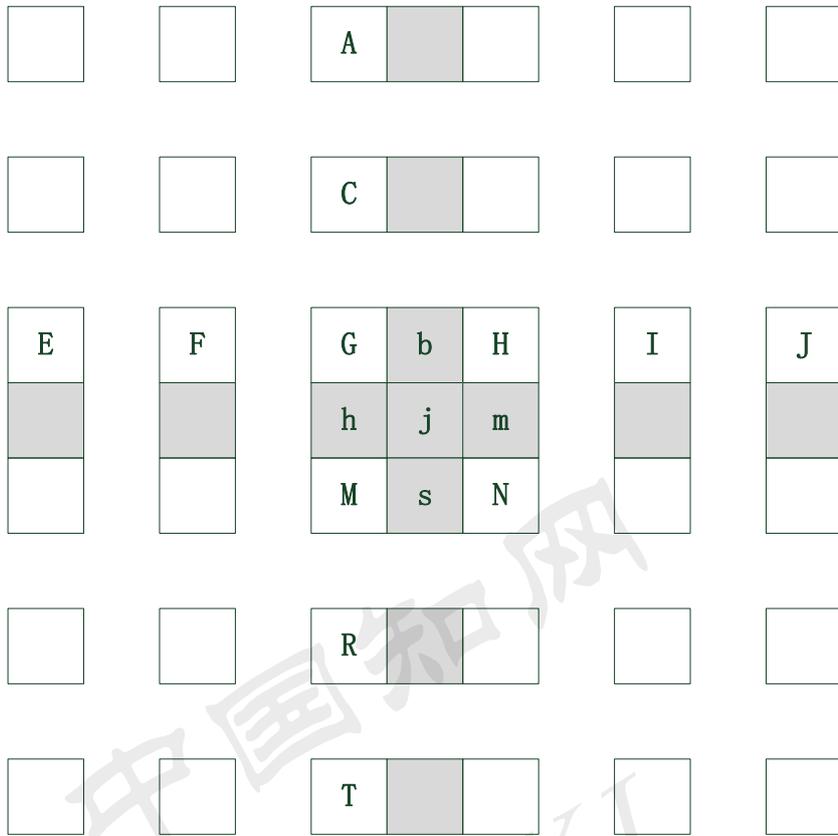


图 3-6 亮度块的半像素位置的插入

同理，半像素点 h 可以通过对 A, C, G, M, R, T 进行差值得到。当得到所有的半像素点后， $1/4$ 像素位置的像素点由线性插补得到^[30]。例如，对于图 3-7 中的 $1/4$ 像素 a ，其差值公式为 3-2：

$$a = \text{round}\left(\frac{G+b}{2}\right) \quad (3-2)$$

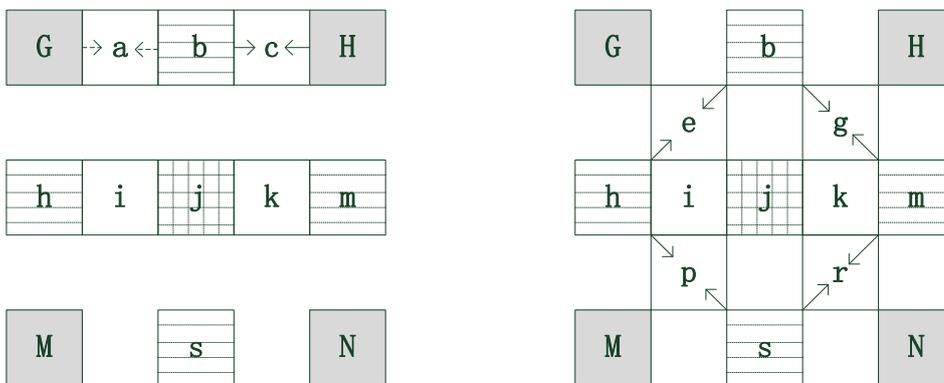


图 3-7 亮度块的 $1/4$ 像素位置的插入

对于对角线上的 $1/4$ 像素 e ，其差值公式为 3-3

$$e = \text{round}\left(\frac{h+b}{2}\right) \quad (3-3)$$

3.2.2 变换与量化

3.2.2.1 整数变换

H.264 标准主要有三种变换模式：对于 16×16 块的帧内预测的宏块中，亮度直流系数的 4×4 矩阵使用哈达玛变换；任何宏块的色度直流系数的 2×2 矩阵使用哈达玛变换；所有其他残差数据的 4×4 块使用基于 DCT 的变换^[31]。这些变换是基于 DCT 的，但是与 DCT 变换又有一些不同。

3.2.2.2 量化

H.264 标准使用一个标量量化器，它将每个图像样点编码映射成较小的数值。标量量化器的原理如公式 3-4 所示：

$$Z_{ij} = \text{round}\left(\frac{Y_{ij}}{Q_{\text{step}}}\right) \quad (3-4)$$

Y_{ij} 是上述变换的一个系数， Q_{step} 是量化步长大小， Z_{ij} 是量化后的系数。 round 表示舍入操作。由于 H.264 标准在量化过程中同时还要完成 DCT 变换中的加速尺度操作，因此公式 3-4 可以描述为公式 3-5：

$$Z_{ij} = \text{round}\left(W_{ij} \cdot \frac{PF}{Q_{\text{step}}}\right) \quad (3-5)$$

根据具体的位置 (i, j) ，PF 可以取值 a_2 ， ab 或者 $b_2/4$ 中的一个。在量化过程中，量化步长 Q_{step} 的大小决定量化器的编码压缩率和图像质量。H.264 标准支持多达 52 个 Q_{step} 值，使用量化参数 QP 进行索引。QP 每增长 6， Q_{step} 就增加一倍。量化步长的广阔范围使得编码器能够灵活准确的控制比特率和图像质量之间的权衡。

3.2.3 熵编码

H.264 标准规定了两种熵编码方式：一种是基于指数哥伦布(Exp-Golomb)编码的变长编码(UVLC)和基于上下文的变长编码(CAVLC)相结合的变长编码(VLC)，另一种是基于上下文的自适应算术二进制编码(CABAC)^[32]。

H.264 标准中，在条带层以上，语法元素被编码成定长的或者可变长度的二进制码字。在条带层及条带层以下，可以采用变长编码或者自适应的算术编码 CABAC。在变长编码中，除了量化系数外均采用指数哥伦布编码。量化系数则采用 CAVLC 编码。CABAC 是一种基于概率模型的统计估算，与变长编码 CAVLC 相比，对同样质量的视频信号编码，CABAC 可以节约大约 10%~15%

的码率^[33]。

3.2.3.1 指数哥伦布编码

因为指数哥伦布编码具有更好的错误恢复能力,因此用来编码量化系数等重要参数。哥伦布编码过程分为两步:第一步将待编码的数据转换为一个中间变量 `code_num`; 第二步将 `code_num` 映射为二进制编码。哥伦布编码按照一个固定的逻辑方式构造:

$$\text{Codeword}=[M \text{ 个 } 0][1][\text{INFO}]$$

其中, INFO 为 M 个 bit 数据。编码器端, 每个码字的构造如下:

$$M = \text{floor}(\log_2[\text{code_num} + 1])$$

$$\text{INFO} = \text{code_num} + 1 - 2^M$$

3.2.3.2 CAVLC 编码

CAVLC 编码的基本思想是对出现概率大的符号分配较短的二进制码, 出现较少的符号使用较长的二进制码。同时, 它利用经过变换和量化后的 4×4 矩阵的特性来实现更高效的编码。

1. 经过变换和量化后的块是典型的稀疏矩阵, 预测残差中含有较多的 0, 这样在 Zig-Zag 扫描之后, 用游程(非零系数前零的数目)和振幅(非零系数值)表示预测残差可以取得较好的压缩效果^[34]。

2. 经过扫描后, 最后的非零系数往往是 ± 1 序列, CAVLC 对它们进行单独编码。

3. CAVLC 利用当前块中的非零系数和周围块中的非零系数的相关性自适应的选择编码当前块中的非零系数的码表。

4. CAVLC 利用不同频域的系数值大小自动选择振幅的码表。

CAVLC 编码的过程大体如下:

- 1) 编码非零系数的数目和拖尾系数的数目;
- 2) 编码每个拖尾系数的符号;
- 3) 编码余下非零系数的幅值;
- 4) 编码最后一个非零系数前零的总数;
- 5) 编码每个非零系数前零的个数。

3.2.3.3 CABAC 编码

H.264 引入了基于内容的自适应二进制算术编码 CABAC, 它除了具有一般算术编码的特点外, 它很好地利用了语法元素数值之间的高阶信息, 使得熵编码的效率得到了进一步提高, 它主要有以下特点:

1. 利用每个语法元素的上下文关系, 根据已编码元素为待编码元素选择

概率模型，既上下文建模^[35]；

2. 根据当前的统计特性自适应的进行概率估算；
3. 使用算术编码，而不是变长编码。

3.3 H.264 解码方案研究与选择

H.264 标准规定了符合 H.264 标准的档次、级别与码流范围，但是并没有规定具体的编解码算法。H.264 标准自 2003 年公布以后，世界各地的各个组织和研究机构都研发出了自己的 H.264 编解码器。这些开源代码在支持 H.264 特性、解码速度和开发难易度等方面不尽相同^[36]。目前流行的开源 H.264 解码器主要有以下 4 种：

- 1) JM: JM 系列是 H.264 标准的官方测试源码，由德国 HHI(Heinrich Hertz Intiut)研究所负责开发，它注重实现 H.264 标准丰富的功能，并没有专门进行优化。因此该源代码的特点是引入各种新特性提高编解码性能，但是结构冗长、复杂度高。适合进行学术研究但是实用性差。
- 2) X264: X264 是由法国巴黎中心学校的中心研究所的一些学生在网上组织发起的，并由众多视频编解码爱好者共同完成的。其目的是实现实用的 H.264 编解码器。X264 摒弃了一些耗时但是对编码性能提高不是很大的一些功能模块，因此其相比较 JM 系列而言，在程序结构和算法性能方面有了提高。X264 实现了 H.264 标准的基本档次编码器的基本功能和另外两个档次的部分功能。但是它还没有实现真正的解码功能。
- 3) T264: T264 是由中国视频编码自由组织联合开发的 H.264 编解码器。T264 和 X264 在程序结构和性能方面有些类似，也是注重实用性，吸收了 JM、X264 的优点。T264 编码输出标准的 H.264 码流，但是其解码只能解码 T264 本身的码流。
- 4) FFmpeg: FFmpeg 是一个集录制、转换、音/视频编解码功能为一体的完整的开源的音频和视频流解决方案。它支持各种音视频标准编解码，还支持各种文件格式(.avi,.mp4,.mkv 等)的解析。FFmpeg 被许多开源项目采用，比如 ffmpeg2theora, VLC, MPlayer, HandBrake, Blender, Google Chrome 等。另外，一些著名的播放软件，例如暴风影音、QQ 影音和 KMPlayer 等，里面也采用了 FFmpeg 的开源代码。FFmpeg 是一个非常好的音视频编解码库，支持的标准非常全面，而且解码速度也很快。

比较以上几个开源的解码器可以发现：**JM** 系列代码结构冗长，只考虑引进新特性提高编解码性能，忽视了效率，编码复杂度极高，适合学术研究，没有实际应用价值。**X264** 和 **JM** 相比，提高了编码速度，但是其实际上只是一个编码器，还没有真正的解码功能。**T264** 的编解码性能都有了很大提高，但是其只能解码 **T264** 本身的码流，具有一定得局限性，通用性不好。通过对以上解码器的研究，针对程序开发上的难以程度、适用场合等做比较后，本次开发决定采用 **FFmpeg** 的解码器为原型，经过适当的裁剪优化后进行移植。

FFmpeg 是一个集录制、转换、音/视频编解码功能为一体的完整的开源的音频和视频流解决方案。**FFmpeg** 是基于 **linux** 开发的，可以在但多数操作系统中编译和使用。它支持 **MPEG**、**DivX**、**MPEG4**、**AC3**、**FLV** 等 40 多种编码和 **AVI**、**MPEG**、**OGG**、**ASF** 等 90 多种解码^[37]。**FFmpeg** 被许多开源项目采用，比如 **ffmpeg2theora**、**VLC**、**MPlayer**、**HandBrake**、**Blender**、**Google Chrome** 等。还有 **DirectShow/VFW** 的 **ffdshow** (external project) 和 **QuickTime** 的 **Perian** (external project) 也采用了 **FFmpeg**。另外，一些著名的播放软件，例如暴风影音、**QQ 影音** 和 **KMPlayer** 等，里面也采用了 **FFmpeg** 的开源代码，但是他们并没有遵守 **LGPL/GPL** 协议，没有公开任何源代码，已经被 **FFmpeg** 组织列入黑名单。

FFmpeg 的核心项目组成主要包括以下几个部分：

- ◆ **Libavcodec**: 包含了 **FFmpeg** 所需要的音视频编解码器(encoder/decoder) 的库，是 **FFmpeg** 的核心部分。为了保证高的可移植性和编解码质量，**libavcodec** 里面好多 codec 都是从头开发的。本文需要进行移植的 **H.264** 解码器就是 **libavcodec** 里面的一部分。
- ◆ **Libavformat**: 包含了各种音视频格式的解析器(demuxer)和产生器(muxer)的库。用于各种音视频封装格式的生成和解析，包括获取解码所需信息以生成解码上下文结构和读取音视频帧等功能。
- ◆ **libavutil**: 包含一些公共的工具函数。该库实现了 **CRC** 校验码的产生、最大公约数、整数开方、内存分配、大端小端格式的转换等功能。
- ◆ **libswscale**: 用于视频场景比例缩放、色彩映射转换等。
- ◆ **libpostproc**: 用于后期效果处理

一般来说，**FFmpeg** 处理视频的大体流程如下：

- 1) 从 **test.264** 文件中打开视频流 **video.stream**
- 2) 从视频流中读取包到帧中

- 3) 如果这个帧不完整, 跳回到 2)
- 4) 对完整帧进行操作
- 5) 跳回到 2)

本文按照 FFmpeg 的解码流程为框架, 对 ffmpeg 中 H.264 解码库进行精简优化, 移植到 android 手机操作系统, 作为 H.264 解码库的原型。

3.4 解码器裁剪与优化

FFmpeg 的开发是基于 linux 系统的, 而 Android 也是以 linux2.6 为内核开发的。因此, FFmpeg 移植到 Android 系统中, 应该可以得到很好的性能。但是, FFmpeg 是一个很大的集音视频编解码为一体的开源解决方案, 对于本文来说, 完全移植没有必要。这样, 就需要对 FFmpeg 进行精简和优化, 从中裁剪分离出 H.264 的核心代码进行移植。对于 FFmpeg 这样一个庞大的源代码复杂的开源方案中找到本文需要的代码, 是一项非常艰难的工作。

3.4.1 FFmpeg 关键数据结构

3.4.1.1 AVFormatContext

在 FFmpeg 代码中, AVFormatContext 是一个贯穿始终的数据结构, 很多函数都用它作为参数。此结构体包含了一个视频流的格式内容。AVFormatContext 中存有 AVInputFormat(or AVOutputFormat 同一时间只能存在一个)和 AVStream、AVPacket 等几个重要的数据结构以及其他一些在解码中可能用到的相关信息, 诸如: duration,file_size,bit_rate 等。

3.4.1.2 AVStream

AVStream 是继 AVFormatContext 之后第二个贯穿始终的结构数据。作为“Stream”, 它包含了“流”这个概念中的一些数据, 比如: 帧率(r_frame_rate)、基本时间计量单位(time_base)、需要编解码的首帧位置(start_time)、持续时间(duration)、帧数(nb_frame)以及一些 ip 信息。其中还包括两个重要的数据结构成员:

```
AVCodecContext *codec;  
void *priv_data;
```

其中, codec 指针保存的是 encoder 或 decoder 结构。priv_data 指针保存的是和具体编解码流相关的数据。根据输入或者输出流的不同, AVStream 结构被封装在 AVInputStream 和 AVOutputStream 结构体中。

3.4.1.3 AVCodecContext

AVCodecContext 是最重要的数据结构之一。保存 AVCodec 指针和与 codec 相关的数据结构。其中包括视频的宽度、高度、采样率等。其中的 codec_type, codec_id 两个变量对于编解码器的匹配来说最为重要。在 H.264 解码中, ffmpeg 先调用 av_open_input_file() 函数, 匹配到文件格式解析器 demuxer, 然后通过 av_open_input_stream() 函数调用 read_header 接口来执行 H264.c 中的 flv_read_header() 函数。在 flv_read_header() 函数内, 根据文件头中的数据, 创建相应的音频流或视频流, 并设置 AVStream 中 AVCodecContext 的正确的 codec_type 值。

3.4.1.4 AVCodec

结构体 AVCodec 中成员函数和变量比较少, 但是很重要。它包含了编解码器的 CodecID, 也就是用哪个 Codec 实现编解码。

3.4.1.5 AVFrame

AVFrame 结构定义如下:

```
typedef struct AVFrame{  
    FF_COMMON_FRAME  
}AVFrame
```

其中 FF_COMMON_FRAME 是以一个宏的形式出现的。由于在编解码过程中, AVFrame 中的数据是要经常存取的。为了加速, 所以采取这样的手段。其中包含关键帧 key_frame、参考帧 referenc、宏块类型 *mb_type 等等。

3.4.1.6 AVPacket

AVPacket 是写入文件的基本单元, 用于保存读取的 packet 数据。在做视频传输、同步、边界等问题时, 可以通过 AVPacket 来解决。

3.4.2 H.264 解码器裁剪优化

本文移植的 H.264 解码器, 主要是以 FFmpeg 中 H.264 的解码器为原型的。FFmpeg 是一个大的开源工程, 它实现当前许多流行的音视频的编解码器。例如 OGG、MPEG-2、MPEG-4、H.263 等。本文仅仅是研究 H.264 解码器, 因此可以将 FFmpeg 中与 H.264 解码器无关的代码去掉, 对其进行深度裁剪, 以节省代码存储空间。

1. 数据结构优化: FFmpeg 设计了统一的接口来说实现其内部对各种编解码器的支持。它通过一些公用的结构体来描述各种编解码器需要用到的资源变量和特性。其中, 许多变量对 H.264 解码器来说是没有用处的。因此, 本文首先针对 H.264 解码器对 FFmpeg 中的结构体进行裁剪优化, 去掉与 H.264 解码器无关和解码过程中用不到, 或者重复使用的变量。例如: 对于 AVFrame

结构体来说，优化如下：

```
typedef struct AVFrame {
    uint8_t *data[4];
    int linesiz[4];
    uint8_t *base[4];
    int key_frame;
    int pict_type;
    int reference;
    int8_t *qscale_table;
    uint32_t *mb_type;
}AVFrame;
typedef struct AVCodecContext{
    int width,height;
    int hurry_up;
    void *priv_data;
    AVFrame *coded_frame;
    int internal_buffer_count;
    void *internal_buffer;
}AVCodecContext;
```

2. 冗余函数模块：参考代码中的一些模块函数是辅助作用的，例如一些打印调试信息的，为了优化需要，可以将这些辅助模块删掉。

表 3-3 冗余模块信息

函数模块	功能描述
TRACE	跟踪打印信息，输出到文件
Check_marker	检查码流丢失数量
ff_printf_debug	打印调试信息

经过对重要数据结构的裁剪优化后，下一步就是对 H.264 解码的核心算法的提取。对于解码基本档次的 H.264 来说，在经过反复研究尝试后，保留 H264.c、utils.c(H264utils.c)、Cabac.c、golomb.c 等几个源文件。

H264.c: H.264 解码器的核心算法实现部分。

H264utils.c: 裁剪了 FFmpeg 的 utils.c 而得到的。主要负责内存分配、管理等操作。

Cabac.c: 自适应的算术二进制编码，是 H.264 采用的一种熵编码方式。

Golomb.c: 哥伦布编码方式，是一种有规则的变长编码方式。在各类视频编码标准中被广泛采用。在 H.264 中采用的 0 阶 golomb 编码。

3.5 解码器编译移植

在完成解码器的裁剪优化后，就需要进行解码器的交叉编译移植工作。

Android 手机操作系统是面向应用,使用面向对象的 java 语言进行应用开发的。因此,想要在 Android 的应用层调用解码库来实现解码功能,还需要对解码库进行封装,利用 java 的本地调用来实现解码器才行。

3.5.1 生成头文件

在 java 层声明本地方法后,需要使用 javah 命令需要生成 JNI 头文件,这里需要注意完整的包名。这里生成了一个头文件,其中包含了需要利用本地 C 实现的方法名。

```

        /* DO NOT EDIT THIS FILE - it is machine generated */
#include <jni.h>
/* Header for class com_android_h264decoder_VideoPlayer */

#ifndef _Included_com_android_h264decoder_VideoPlayer
#define _Included_com_android_h264decoder_VideoPlayer
#ifdef __cplusplus
extern "C" {
#endif
/*
 * Class:      com_android_h264decoder_VideoPlayer
 * Method:     InitDecoder
 * Signature:  ()I
 */
JNIEXPORT jint JNICALL Java_com_android_h264decoder_VideoPlayer_InitDecoder
    (JNIEnv *, jobject);

/*
 * Class:      com_android_h264decoder_VideoPlayer
 * Method:     UninitDecoder
 * Signature:  ()I
 */
JNIEXPORT jint JNICALL Java_com_android_h264decoder_VideoPlayer_UninitDecoder
    (JNIEnv *, jobject);

/*
 * Class:      com_android_h264decoder_VideoPlayer
 * Method:     DecoderNal
 * Signature:  ([BI[B)I
 */
JNIEXPORT jint JNICALL Java_com_android_h264decoder_VideoPlayer_DecoderNal
    (JNIEnv *, jobject, jbyteArray, jint, jbyteArray);

```

```

#ifdef __cplusplus
}
#endif
#endif

```

3.5.2 本地 C 实现

本文主要封装实现了 3 个接口，分别是：

InitDeocoeer()：初始化解码器，主要完成解码需要的内存分配等功能。

DeceoderNal()

```

{
byte[] input    //输入缓冲区
int nalLen     //缓冲区大小
byte[] output  //输出缓冲区
}

```

UninitDecoder()：释放解码器有关的资源

3.5.3 编译本地方法

要编译移植 H.264 解码器到 Android 中。就需要用到 Android 提供的 NDK 工具^[38]。Android 的 NDK 是一系列开发工具的集合，集成了交叉编译器，并提供 android.mk 文件隔离 CPU、平台、ABI 等的差异，开发人员只要编写自己的 android.mk 文件，就可以实现交叉编译，产生 .so 文件和 java 打包供开发人员使用。这样，就为开发人员使用 C/C++ 开发 Android 提供了一个简单而有效地途径。

Android 的 NDK 需要使用 gcc 进行交叉编译^[39]。因此，要使用 NDK 进行交叉编译，可以有两种思路配置编译环境。一种是使用纯 linux 环境进行编译开发。如果工作在 windows 环境下而又不想使用 linux 环境，那么可以安装 cygwin 来模拟出一个类 linux 的环境来进行编译。因为 Android 是以 linux2.6 内核开发的，同时为了避免因为环境问题而出现不必要的麻烦，本文直接在 Ubuntu8.1 环境下进行 NDK 的交叉编译。

本文使用的 NDK 版本是 android-ndk-r4-linux 最新版本。编译时需要配置的 android.mk 文件编写如下：

```

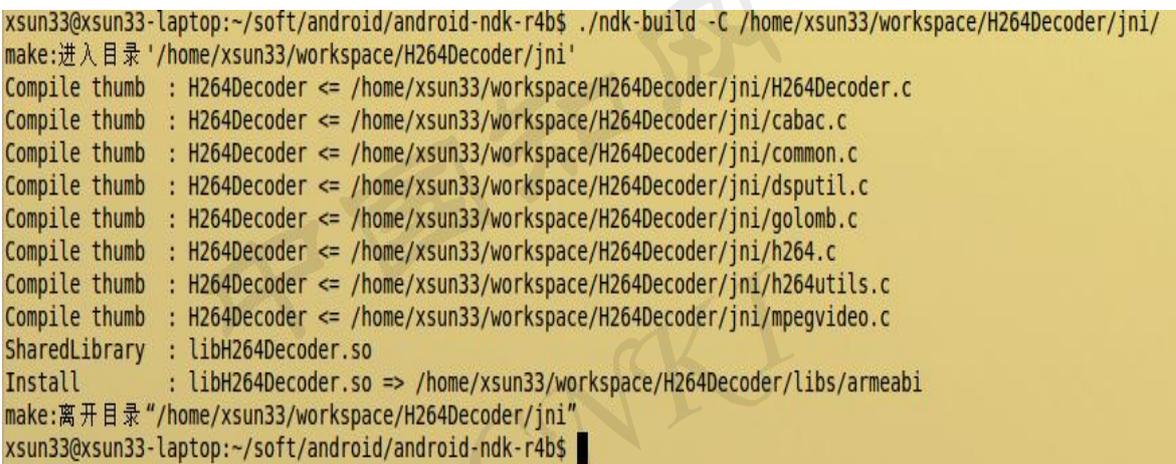
LOCAL_PATH :=$(call my-dir)
include $(CLEAR_VARS)
LOCAL_MODULE :=H264Decoder

```

```
LOCAL_SRC_FILES :=H264Decoder.c \
                cabac.c \
                common.c \
                dsputil.c \
                golomb.c \
                h264.c \
                h264utils.c \
                mpegvideo.c
```

```
include $(BUILD_SHARED_LIBRARY)
```

执行编译命令：`./ndk-build -C /H264Decoder/jni/`，如果编译成功，会出现如下图所示界面：



```
xsun33@xsun33-laptop:~/soft/android/android-ndk-r4b$ ./ndk-build -C /home/xsun33/workspace/H264Decoder/jni/
make:进入目录 '/home/xsun33/workspace/H264Decoder/jni'
Compile thumb : H264Decoder <= /home/xsun33/workspace/H264Decoder/jni/H264Decoder.c
Compile thumb : H264Decoder <= /home/xsun33/workspace/H264Decoder/jni/cabac.c
Compile thumb : H264Decoder <= /home/xsun33/workspace/H264Decoder/jni/common.c
Compile thumb : H264Decoder <= /home/xsun33/workspace/H264Decoder/jni/dsputil.c
Compile thumb : H264Decoder <= /home/xsun33/workspace/H264Decoder/jni/golomb.c
Compile thumb : H264Decoder <= /home/xsun33/workspace/H264Decoder/jni/h264.c
Compile thumb : H264Decoder <= /home/xsun33/workspace/H264Decoder/jni/h264utils.c
Compile thumb : H264Decoder <= /home/xsun33/workspace/H264Decoder/jni/mpegvideo.c
SharedLibrary : libH264Decoder.so
Install       : libH264Decoder.so => /home/xsun33/workspace/H264Decoder/libs/armeabi
make:离开目录 "/home/xsun33/workspace/H264Decoder/jni"
xsun33@xsun33-laptop:~/soft/android/android-ndk-r4b$
```

图 3-8 编译成功截图

编译成功后，会在工程目录下自动生成一个 `/libs/armeabi` 文件夹，编译得到的动态库 `H264Decoder` 会安装在这个文件夹下。然后在工程中的 `java` 代码部分添加如下代码：

```
static {
    System.loadLibrary("H264Decoder");
}
```

这样，就完成了解码器的交叉编译和移植工作。

3.6 本章小结

本章在分析了 H.264 的算法和几种 H.264 开源解码器的基础上，选定解码速度较快的 FFmpeg 解码器为原型进行移植。重点对 FFmpeg 的解码流程进行跟踪分析，从中分离出 H.264 解码的核心代码并精心优化，移植到 Android 平台系统，作为本文解码库的核心部分。

第 4 章 解码器客户端实现

4.1 开发环境搭建

本课题需要在 Android 模拟器上开发实现 H.264 解码器。同时，为了更好的调试代码和交叉编译，需要使用的 gcc 编译工具。综合考虑这几个因素，本文决定采用 linux 下的 EclipseIDE 集成开发环境为基础。同时加上 Android 提供的 SDK 和 NDK 开发工具，在 Eclipse 下构建 Android2.1 的模拟器。主要需要的软件资源包括：

系统环境：Ubuntu8.1

SDK：Android SDK 2.1

NDK：Android NDK r4 for linux

开发环境：Eclipse3.4+jdk1.6+Android ADT

开发环境搭建步骤：

- ◆ 第一步是安装 java SDK。要进行 Android 的开发，安装的 jdk 必须是 1.5 或者 1.5 以上版本。本课题安装的是 jdk 1.6 linux 版本。
- ◆ 第二步是安装 Eclipse 开发环境。下载 Eclipse 并解压即可。注意需要的 Eclipse 版本必须是 3.3 及其以上版本。本文采用 Eclipse3.4。
- ◆ 下载 Android ADT 插件，并安装到 Eclipse 中。
- ◆ 下载并解压 Android NDK。
- ◆ 下载并解压 Android SDK。配置 Eclipse 中 android SDK 的路径，创建 Android 2.1 模拟器，运行一个 Hello World 程序，开发环境搭建成功。

4.2 解码器整体设计

解码器的整体设计可以分成两部分，一部分是视频数据的解码部分，主要用 C 语言来实现，采用 Android NDK+C 的实现机制。另外一部分是视频的显示部分，主要采用 Android 提供的组件来实现，采用 Android SDK+Java 的实现机制。而这两部分的集合，则是通过 java 提供的 jni 机制来实现 Java 和 C 语言之间的通信^[40,41]。整个解码器的框架如图 4-1 所示。

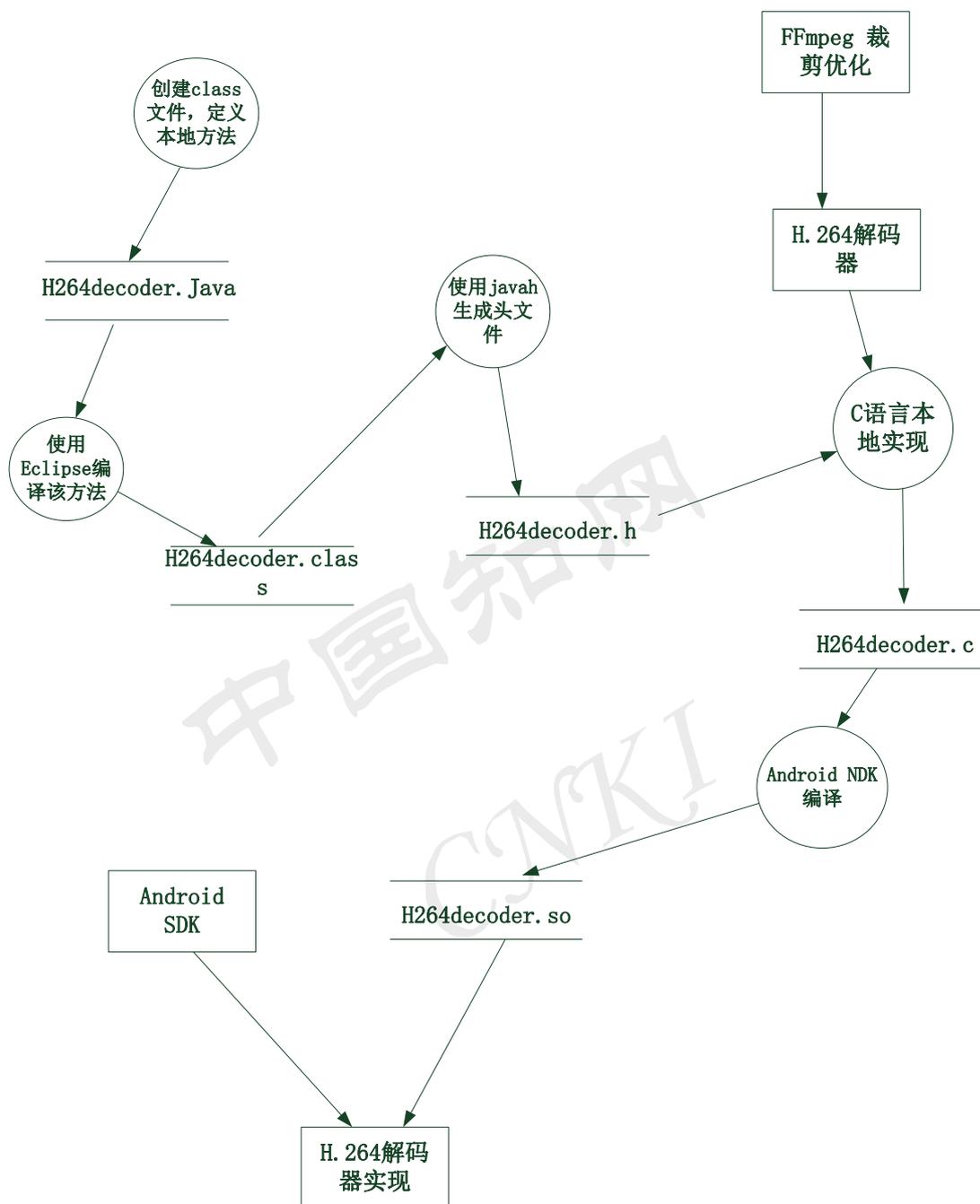


图 4-1 解码器整体实现框架图

4.3 解码流程

整个解码的流程可以分成三个功能模块：前端码流处理、H.264 解码和后端视频显示。

- ◆ 前端码流处理：主要负责文件的读取，从码流中分割出 NAL 然后交给底层进行解码处理。

- ◆ H.264 解码：整个解码的核心部分，通过本地 C 语言的实现和解码库对码流数据进行处理，完成 H.264 解码实现图像重建。
- ◆ 后端视频显示：接收 H.264 解码模块解码后的视频数据，在 Android 客户端进行显示。

通过分析三个模块的功能可知，H.264 解码模块是最耗费资源的。本模块通过移植的 H.264 解码库来实现解码。另外两个模块则在 Android 的 java 层进行实现。

H.264 视频标准为了更好的适应网络传输的特性，采用了分层设计的思想既视频编码层 VCL 和网络提取层 NAL。H.264 的解码部分有包括前端码流处理和 H.264 解码两个功能模块。其中，前端码流处理主要完成从码流中分割出 NAL，这部分功能由 java 层实现。在 java 层，采用 `java.io.FileInputStream` 类来实现码流的读取。读取文件后，我们在 java 层从 H.264 码流中分割出 Nal，然后交给底层的 C 来实现实时解码。Nal 的分割可以在底层通过 C 来实现，但是这样的话，就会出现很多问题。上层 java 层每次应该送多少数据给下层呢？如果一次送的太多，底层可能解码出好几帧的数据，但是通知到界面层后，只能显示一帧的数据，这样就造成了丢帧的现象。若是送的数据太少，底层就无法进行实质的解码，可能需要上层等待多次传送后，才能解码出一帧完整的数据通知上层显示。因此，综合考虑之后，决定在 java 完成 Nal 的分割。整个视频解码器的工作流程如下：

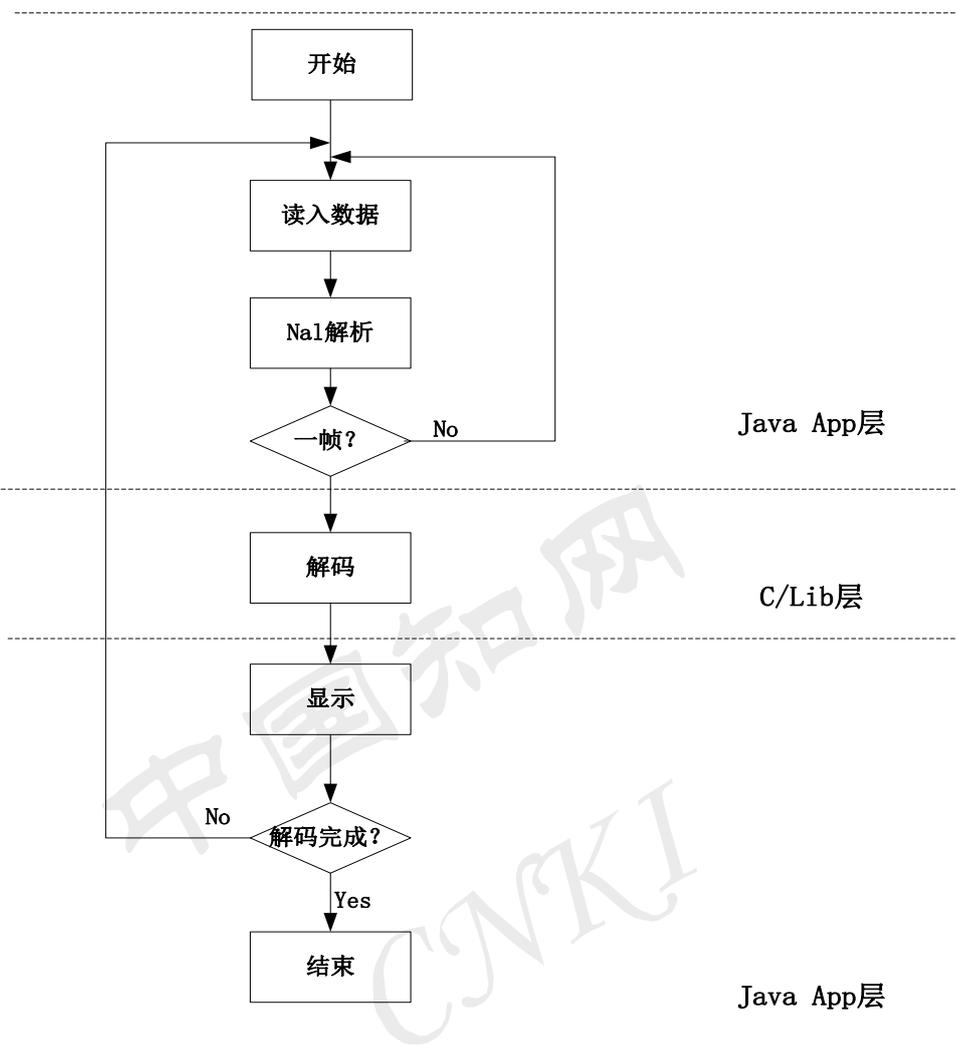


图 4-2 视频播放器工作流程

完成 Nal 的分割之后，将数据交给底层解码器处理，对于具体视频数据，解码流程如下：

- 1) 打开解码器：avcodec_open()
- 2) 为解码帧分配内存：avcodec_alloc_frame()
- 3) 不停的从码流中提取帧数据：av_read_frame()
- 4) 对于视频帧调用：avcodec_decode_frame()
- 5) 解码完成，释放解码器：avcodec_close()

4.4 视频显示

4.4.1 YUV 到 RGB565 色彩转换

本文的视频显示采用的是 Android 提供的 Bitmap 类显示 BMP 图像，模拟

实现视频接口。H.264 标准支持 4:2:0 的逐行和隔行的视频采样格式。因此，H.264 码流解码出来的是 4:2:0 的 YUV 格式。为了能够在应用层完整的显示图像，需要完成色彩空间从 YUV 到 RGB 的转换。

在 RGB 颜色空间中，每个像素值使用红、绿、蓝三个数值来表示。它们是组成有色光的三种基本色：任何颜色都可以通过红、绿、蓝以一定得比例组合来重现^[42]。因此，RGB 颜色空间可以使用 8bit 来表示每一种色彩空间。但是，人类的视觉系统对色度的敏感程度低于亮度。而在 RGB 空间里，三种颜色分量的重要性相同，所以也必须以相同的度量来存贮它们，但事实上我们可以通过提高亮度的精度，降低色度的精度来更有效地表示颜色空间^[43]。

通常， YC_bC_r 色彩空间和它的变形(YUV 色彩空间)是用于有效表示彩色图像的对色彩空间度量的方法。Y 指的是亮度(Luma)，是不同权重的 R、G、B 的平均：

$$Y = k_r R + k_g G + k_b B \quad (4-1)$$

k 是权重。色彩信息可以表示为色差(Chroma)，每一个色差表示了 RGB 与 Y 的差值：

$$\begin{aligned} C_b &= B - Y \\ C_r &= R - Y \\ C_g &= G - Y \end{aligned} \quad (4-2)$$

因为是 $C_b + C_r + C_g$ 一个常数,所以我们只需要存储或者传输其中的两个，第三个可以计算出来。在 YC_bC_r 空间只有亮度 Y 和蓝色、红色的色差传输。 YC_bC_r 相对于 RGB 而言，有一个巨大的优点就是我们可以用比 Y 更低的分辨率来存储 C_bC_r ，因为人类视觉系统对色度的敏感程度低于亮度，这样就可以大量减少数据的同时对视觉质量没有影响^[44]。

将 RGB 图像转换成 YC_bC_r 图像是为了减少存储和传输数据，在显示图像之前，通常还要在转换成 RGB。公式 4-3 和 4-4 给出了变换和逆变换的公式。

$$\begin{aligned} Y &= k_r R + (1 - k_b - k_r)G + k_b B \\ C_b &= \frac{0.5}{1 - k_b} (B - Y) \\ C_r &= \frac{0.5}{1 - k_r} (R - Y) \end{aligned} \quad (4-3)$$

逆变换公式：

$$R = Y + \frac{1 - k_r}{0.5} C_r$$

$$G = Y - \frac{2k_b(1-k_b)}{1-k_b-k_r} C_b - \frac{2k_r(1-k_r)}{1-k_b-k_r} C_r \quad (4-4)$$

$$B = Y + \frac{1-k_b}{0.5} C_b$$

国际电信联盟(ITU-R)推荐的标准中定义的 $k_b=0.114$, $k_r=0.299$ 。带入上面逆变换公式, 得到本文需要的色彩空间转换公式:

$$\begin{aligned} R &= Y + 1.402C_r \\ G &= Y - 0.344C_b - 0.714C_r \\ R &= Y + 1.772C_b \end{aligned} \quad (4-5)$$

以上的转换, 需要进行浮点运算, 而且需要对图像中每个像素点都要进行颜色空间的转换, 计算量复杂。因此, 为了更好的提高性能, 对算法进行优化, 将浮点运算转换为整数运算。首先, 对 0.402、0.344、0.714 和 0.772 取分数近似值: 103/256、88/256、183/256 和 198/256, 这样上面的公式就可以通过整数运算和移位运算来提高效率, 优化后的公式可以表示为:

$$\begin{aligned} R &= Y + C_r + (103 * C_r) >> 8 \\ G &= Y - (88 * C_b) >> 8 - (183 * C_r) >> 8 \\ R &= Y + C_b + (198 * C_b) >> 8 \end{aligned} \quad (4-6)$$

基于上面阐述的颜色空间转换公式, 通过在 C 语言层的两个函数:

```
void CreateYUVTab()
```

```
void DisplayYUV(unsigned int *pdstl, unsigned char *y, unsigned char *u,
unsigned char *v, int width, int height, int src_ystride, int src_uvstride, int
dst_ystride)实现颜色空间 YUV 到 RGB 的转换。
```

4.4.2 RGB 数据显示

颜色空间转换后的数据, 传给 java 层。视频的显示功能是通过 android.graphics.Bitmap 类, 完成 RGB 空间到 Bitmap 的映射, 然后在 android.view.View 里面, 重载 onDraw()方法, 然后用 canvas.drawBitmap()方法, 将图片画在屏幕上。

```
@Override protected void onDraw(Canvas canvas){
super.onDraw(canvas);
VideoBit.copyPixelsFromBuffer(buffer);
canvas.drawBitmap(VideoBit,0,0,null);
}
```

为了方便显示视频图像, 自己编写视频显示空间 VideoPlayer 类, 实现视频的解码显示功能。该类继承自 android.view.View 并集成了接口 Runnable

来实现多线程^[45,46,47]。对外提供 void PlayVideo(String file)、void StopVideo() 和 void PauseVideo()三个简单的接口，实现基本的视频播放功能。该类构造如下：

```
public class VideoPlayer extends View implement Runnable{
    /*重载构造方法*/
    public VideoPlayer(Context context,Attrubits atu){
    }
    /*重写 Ondraw()函数，实现视频显示*/
    @Override protected void onDraw(Canvas canvas){
    super.onDraw(canvas);
    VideoBit.copyPixelLsFromBuffer(buffer);
    canvas.drawBitmap(VideoBit,0,0,null);
    }
    /*播放*/
    public void PlayVideo(String file){
    }
    /*暂停*/
    public void PauseVideo(){
    }
    /*停止*/
    public void StopVideo(){
    }
}
```

4.5 解码器测试结果

本文完成的所有工作都是在 Android2.1 模拟器下进行的，并在真机条件下进行了测试。解码器采用的是应用比较广泛、解码速度比较快的 FFmpeg 解码器为原型。解码后的视频画面截图如图 4-3 所示(模拟器画面截图)。

用本文设计的 H.264 纯软件解码系统对 H.264 码流进行测试。采用的测试序列是 QVGA(320X240)格式的一段视频序列。分别在 PC 机的模拟器和摩托罗拉 ME511 真机下进行测试。测试环境及结果如下：

操作系统：Ubuntu8.0

模拟器：Android2.1 模拟器

PC 机配置：CPU 2.0GHZ RAM 2.0GB

测试结果如下表 4-1 所示：

表 4-1 H.264 解码器在模拟器上的解码速率

视频序列	格式	帧数	解码时间	帧率
Butterfly	QVGA	104	11s	10fps

摩托罗拉 ME511:

操作系统: Android2.1

硬件配置: CPU 528MHZ, RAM 256M

测试结果如下表 4-2 所示:

表 4-2 H.264 解码器在真机 ME511 上的解码速率

视频序列	格式	帧数	解码时间	帧率
Butterfly	QVGA	104	5s	21fps

从上面的测试结果可以看出, 在 Android2.1 模拟器移植的 H.264 解码器, 对于 320×240 大小的 H.264 码流的解码速率可以达到 10fps 左右, 而在真机上的解码速率可以达到 21fps, 基本上满足了实时性的要求, 可以适应实时监控的需要。因此本文移植的解码器, 在将来的 3G 手机视频等监控领域的应用性是可行的。

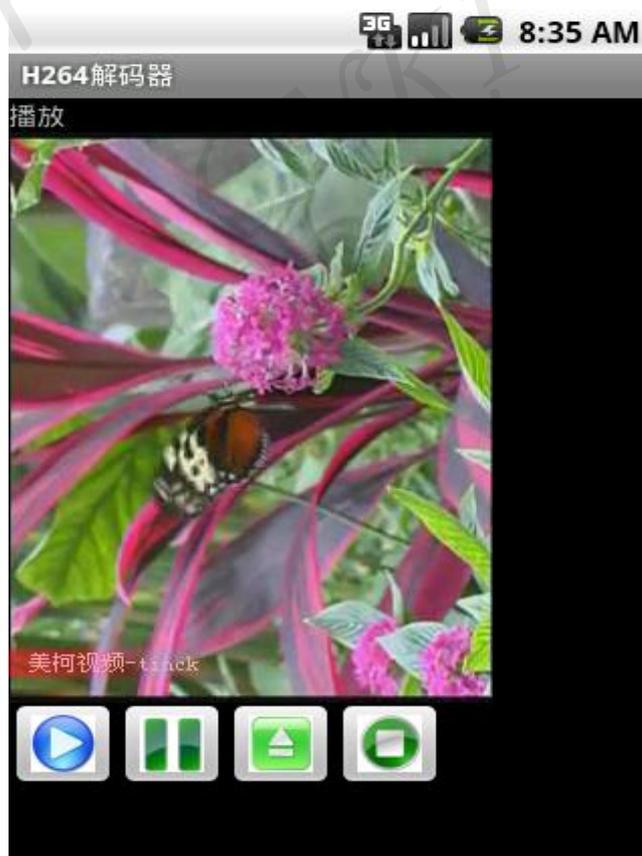


图 4-3 解码器截屏

4.6 本章小结

在移植了 H.264 解码器后,通过 Android 提供的一些基本类和 API 接口的基础上,继承和封装实现了要完成本论文解码器所需要的 VideoPlayer 类,成功的完成了 H.264 的视频解码和播放显示,解码后画面清晰,播放基本流畅。

中国知网
CNKI

结 论

H.264/AVC 视频编解码标准是 ITU-T 和 ISO/IEC 联合发布的视频编码标准,它具有优于 H.263 和 MPEG-4 的压缩性能,可适应更高图像质量和低码率的应用需求。H.264 在吸取了以前的视频编码标准的优点的基础上,采用了一系列新的压缩编码技术,在追求更高的编码效率的同时,提供了很好的视频质量,在无线通信领域和视频监控领域有着广阔的应用前景。

Android 智能手机平台是 Google 于 2007 年发布的最新款的智能手机操作系统,短短的几年时间,Android 凭借其开源的特性,已经在智能手机领域占据了重要的位置。随着 3G 时代的到来,Android 手机的前景不可限量。一时间,Android 成为目前研究最热的智能手机平台之一。因此,本课题结合 Android 智能手机平台,开发手机视频播放系统,设计和实现了基于 Android 操作系统的 H.264 视频播放器。本文完成的主要工作有:

本文研究的主要工作如下:

(1) 深入研究和分析 H.264 视频编码标准,以开源的 FFmpeg 提供的 H.264 解码器为原型,分析了 H.264 解码器中的熵编码、反量化、帧间预测、帧内预测等模块。并对 FFmpeg 解码器进行裁剪优化,从中提取 H.264 解码的核心部分,成功移植到 Android 平台。

(2) 对最新的 Android 智能手机平台进行深入的研究和学习,利用其提供的控件和 API 接口,继承和封装实现了解码功能,设计实现了 H.264 视频播放器,在 Android2.1 模拟器上仿真实现解码显示功能,并在真机下进行了测试。

本文实现了设计要求,但是仍然有许多地方需要改进:

(1) 视频显示模块中,视频画面的大小只能以原始大小进行显示,还无法实现自动缩放功能。

(2) 目前,本课题实现的解码器只能播放本地视频文件,下一步可以进一步改进,实现接收远程视频流,实现远程实时监控、视频会话等功能。

(3) 对 H.264 算法进一步进行优化,提高解码效率。

参考文献

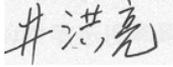
1. ITU-T H.264建议书. 国际电信联盟. 2005,3:2-5
2. Iain E. G. Richardson. H.264 and MPEG-4 Video compression[M]. England: John Wiley & Sons Ltd. 2003:16-18
3. 基于ARM9的H.264解码器的优化与实现. 李辉武[硕士学位论文]. 广东工业大学. 2009,5:4-5
4. Microsoft Tips the Scale in Favor of HTML5 and H.264 Posted by Abel Avram on Apr 30, 2010
5. 刘洋. 超越PC盈利模式: Android打造全新业务平台. 电子设计技术. 2010,17(2):60-66
6. 蔡康, 李洪, 朱英军. 3G网络建设与运营. 人民邮电出版社. 2007,3:4-6
7. 李相周. 流媒体技术及其在3G移动通信中的应用. 通信世界. 2006,5:8-10
8. 彭海涛, 钟锡昌. 嵌入式操作系统在智能手机中的应用. 半导体技术. 2002,16(2):38-40
9. 杨常青, 彭木根. Symbian S60手机程序开发与实用教程. 机械工业出版社 2007,7:8-11
10. 张波, 高朝勤, 杨越. Android基础教程. 人民邮电出版社 2009,8:2-4
11. E2ECloud工作室. 深入浅出Google Android. 人民邮电出版社. 2009,10:2-4
12. <http://it.sohu.com/20101027/n276562747.shtml>
13. Yong-Cai Pan, Wen-chao Liu, LiXiao. Development and research of Music Player Application Based on Android. Communications and Intelligence Information Security. 2010,10:23-25
14. 靳岩, 姚尚朗. Android开发入门与实战. 人民邮电出版社. 2009,9:133-138
15. Tapas Kumar Kundu, Kolin Paul. Android on Mobile Devices: An Energy Perspective. Computer and Information Technology(CIT). 2010,10:2421-2426
16. Thomas Wiegand, Gary J.Sullivan, Senior Member. Overview of the H.264/AVC Video Coding Standard. IEEE 2003,13(7):560-576
17. H.264 and MPEG-4 Video Compression Video Coding for Next-generation Multimedia Iain E.G.Richardson The Robert Gordon University, Aberdeen, UK. 2003,10:16-20
18. Gary J.Sullivan, Pankaj Topiwala, Ajay Luthra. The H.264/AVC Advanced Video Coding Standard: Overview and Introduction to the Fidelity Range Extensions[S]. 2004,8:56-63

19. 毕厚杰. 新一代视频压缩编码标准. 人民邮电出版社. 2005,5:189-200
20. ITU-T. ITU-T Recommendation H.264, Advanced video Coding for Generic Audiovisual Services. 2005,5
21. 欧阳合, 韩军. H.264和MPEG-4视频压缩. 国防科技大学出版社. 2004:17-21
22. Hans cycon, Thomas Schmidt, Gabriel Hege, Matthias Wahlisch, Detlev Marpe, Mark Palkow. Peer-to-peer videoconferencing with H.264 software codec for mobiles. World of Wireless, Mobile and Multimedia Networks. 2008,8:619-622
23. 谢涛. 基于视频压缩标准H.264的研究应用[硕士学位论文]. 中南民族大学. 2009,5:20-22
24. 余姚明, 查日勇, 黄磊等. 图像编码标准:H.264技术. 人民邮电出版社. 2006,10:5-10
25. 田尊华, 程刚. 视频压缩宝典. 人民邮电出版社. 2009,3:38-43
26. Chang-Hong Fu, Yui-Lan Chan, Wan-Chi Siu. Efficient motion estimation in H.264 reverse transcoding. International Conference on Image Processing. March 2007:317-320
27. ITU Telecom. Standardization Sector of ITU. Andvaneed Video Coding for generic audio visual services[C].ITU-T Recommendation H.264. 2005,3
28. Panos Nasiopoulos, Anthony Joch, Faouzi Kossentini. Overview and Performance Evaluation of The ITU-T draft H.264 VideoCoding Standard.Proceedings of SPIE Of Digital Image Processing. 2001
29. 王建, 朱秀吕. H.264中亚像素运动估计算法的优化[J]. 南京邮电学院学报 2005,25(4):48-51
30. Huangqiang Zeng, Canhui Cai, Sanjit K.Mitra. Fast motin estimation for H.264[J]. Signal Processing:Image Communication, 2008,2:2-7
31. Chou-Chen Wang, Chi-Wei Tung, Hsiang-Chun Wang, Ruei-Hung Chiou. Efficient algorithm for early detecting all-zero DCT blocks in H.264 video encoding. Information Science Signal Processing and their Applications(ISSPA). 2010,3:476-479
32. 李振德. 基于H.264编码标准的远程视频采集系统的设计与实现. 湛江海洋大学学报. 2006,26(3):78-81
33. Michael Horowitz, Anthony Joch, Faouzi Kossentini. H.264/AVC baseline profile decoder complexity analysis. Circuits and Systems for Video Technology, IEEE. 2003,13(7):704-716

34. 范斐斐. 基于OMAP5912的H.264编解码器的实现[硕士学位论文]. 武汉理工大学. 2008,5:33-36
35. 闵玲, 李方, 何小海. CABAC在H.264/AVC中的应用. 信息与电子工程. 2006,8:269-274
36. Peter Lee. H.264开源解码器测评. 2006,5
37. FFmpeg开发工程组. FFmpeg开发手册. <http://www.ffmpeg.com.cn>
38. Xiangling, FuXiangxiang, WuMaoqiang, SongMianChen. Research on Audio/Video Codec Based on Android. Wireless communications Networking and Mobile Computing(WiCOM). 2010,9:1-4
39. Richard Petersen. Linux Programmer's Reference. Mc Graw-Hill Companies[M]. 2000,13:170-187
40. 李钟慰, 马文强, 陈丹丹. Java从入门到精通. 清华大学出版社. 2008,9:257-260
41. Lingyan Bi, Weining Wang, Haobin Zhong, Wenxuan Liu. Design and Application of Remote Control System Using Mobile Phone with JNI Interface. Embedded Software and Systems Symposia. 2008,6:416-419
42. Iain E.G.Richardson 欧阳和,韩军. 视频编解码器设计—开发图像与视频压缩系统. 国防科技大学出版社. 2005,1:11-17
43. Benjamin Gordon, Navin Chaddha, teresaH. A Low_power Mutiplierless YUV to RGB Converter Based on Human Vision Perception. VLSI Signal Processing. 1994,10:408-417
44. Sang Heon Lee, Nam Ik Cho. Intra prediction method based on the linear relationship between the channels for YUV4:2:0 intra coding. Image Processing(ICIP). 2009,9:1037-1040
45. 张白一, 崔尚林. 面向对象程序设计:Java. 西安电子科技大学出版社. 2006,1:122-130
46. Chun-Mok Chung, Shin-Dug Kim. A dualthreaded java processor for java multithreading. Parallel and Distributed Systems. 2002,8:693-700

哈尔滨工业大学硕士学位论文原创性声明

本人郑重声明：此处所提交的硕士学位论文《基于 Android 的 H.264/AVC 解码器的设计与实现》，是本人在导师指导下，在哈尔滨工业大学攻读硕士学位期间独立进行研究工作所取得的成果。据本人所知，论文中除已注明部分外不包含他人已发表或撰写过的研究成果。对本文的研究工作做出重要贡献的个人和集体，均已在文中以明确方式注明。本声明的法律结果将完全由本人承担。

作者签名：  日期： 2010 年 12 月 31 日

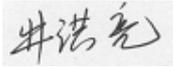
哈尔滨工业大学硕士学位论文使用授权书

《基于 Android 的 H.264/AVC 解码器的设计与实现》系本人在哈尔滨工业大学攻读硕士学位期间在导师指导下完成的硕士学位论文。本论文的研究成果归哈尔滨工业大学所有，本论文的研究内容不得以其它单位的名义发表。本人完全了解哈尔滨工业大学关于保存、使用学位论文的规定，同意学校保留并向有关部门送交论文的复印件和电子版本，允许论文被查阅和借阅，同意学校将论文加入《中国优秀博硕士学位论文全文数据库》和编入《中国知识资源总库》。本人授权哈尔滨工业大学，可以采用影印、缩印或其他复制手段保存论文，可以公布论文的全部或部分内容。

本学位论文属于（请在以上相应方框内打“√”）：

保密□，在 _____ 年解密后适用本授权书

不保密

作者签名：  日期：2010 年 12 月 31 日

导师签名：  日期：2010 年 12 月 31 日

致 谢

光阴似箭，转眼间两年半充实而丰富的研究生生活就要结束了。回顾这两年半的研究生生活，忘事仿佛就在眼前。在我的毕业论文完成和答辩之际，谨向在这研究生生活学习中关心和支持我的各位老师和同学表示深深的敬意和诚挚的谢意。

首先，我要感谢我的导师王鸿鹏教授。王老师学识渊博、治学严谨。他在我的研究生生活学习期间，对我关怀备至，帮我解答许多学习和生活上的问题。王老师注意培养我严谨的治学态度和踏实的学习作风，在科研方面对我严格要求，使我掌握了基本的研究方法。王老师渊博的知识、严谨的治学态度、高尚的风格和对科学的不倦追求的精神给我留下了深刻的印象，必将对我今后的学习和生活产生深远的影响，鞭策我不断学习和进步。

同时，要感谢实验室的秦阳老师，秦老师工作认真，对学生高度负责。她严谨的治学精神、实事求是的作风都给我深刻的印象，秦老师对事业充满热情的投入和勤奋工作的态度不时的感染着我，教育着我，激励着我在学习和生活中不断进取。

感谢计算机学院的每一位领导和老师营造了这么好的学习环境，老师们给予我的帮助和关心，我将铭记于心，永生不会忘怀。

感谢移动计算技术研究中心的同学们，感谢你们在工作学习中给予我的帮助，尤其谢谢许志敏和张巨同学，从你们身上我看到了真正的无私、大度、乐于助人。

最后，感谢我的父母和朋友们，正是他们的无微不至的关心与鼓励才使我有不断前进的动力，让我顺利完成学业。