密级：　　保密期限：

北京郵電大學

硕士研究生学位论文

题目：无线多媒体传感网络关键技术的研究与实现

学号：105710

姓名：钱乐

专业：计算机科学与技术

导师：李文生

学院：计算机学院

2013 年 1 月 2 日

独创性（或创新性）声明

本人声明所呈交的论文是本人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢中所罗列的内容以外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得北京邮电大学或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

申请学位论文与资料若有不实之处，本人承担一切相关责任。

本人签名：_____ 日期：_____

关于论文使用授权的说明

学位论文作者完全了解北京邮电大学有关保留和使用学位论文的规定，即：研究生在校攻读学位期间论文工作的知识产权单位属北京邮电大学。学校有权保留并向国家有关部门或机构送交论文的复印件和磁盘，允许学位论文被查阅和借阅；学校可以公布学位论文的全部或部分内容，可以允许采用影印、缩印或其它复制手段保存、汇编学位论文。

本人签名：_____ 日期：_____

导师签名：_____ 日期：_____

无线多媒体传感网络关键技术的研究与实现

摘　要

从 20 世纪 40 年代电子计算机的诞生到 60 年代数据库、多媒体技术等应用的出现、再到 90 年代前后 Internet 的兴起，信息技术已经逐步融入社会的各个领域并扮演着十分重要的角色。而如今无线传感器网络（Wireless Sensor Networks，WSNs）的出现或将引领信息技术的再一次飞跃，从而进一步拉近人类与物理世界的距离。无线多媒体传感器网络（Wireless Multimedia Sensor Networks, WMSNs）作为 WSNs 的重要分支，融合了 WSNs 技术和多媒体技术，在获取图像、声音、视频等更为丰富的信息的同时，也相应地对研究工作提出了更高的要求。本文选择 WMSNs 中两个重要方面即多媒体数据的采集和传输进行研究与实现，此项工作能够为 WMSNs 的进一步实用化做出贡献。

论文总结了 WSNs 和 WMSNs 的发展前景和国内外研究现状，并在此基础上说明了论文工作的重要意义；描述了本文研究对象所处系统的总体框架和开发平台的搭建过程；详细阐述了对 WMSNs 部分关键技术的研究与实现，包括采集模块和传输模块的设计与编码实现。采集模块有关的工作包括 Linux 内核源码的修改、OV3640 控制模块的实现、H.264 视频编码参数的优化以及音视频数据采集的实现。传输模块有关的工作包括 LT 前向信道编码的实现、基于 Gilbert 模型的信道丢包率统计模块的实现以及最优分组长度实时反馈机制的实现；最后，对上述工作成果进行了系统测试，测试结果表明所实现的采集模块和传输模块在功能上和性能上均满足设计要求。

关键词　WMSNs，S3C6410，数据采集，数据传输，LT 编码

RESEARCH AND IMPLEMENTATION OF KEY TECHNOLOGIES OF WIRELESS MULTIMEDIA SENSOR NETWORKS

ABSTRACT

With the birth of computer in the 1940s, the emergence of applications such as databases and multimedia technologies in the 1960s and the rise of Internet in the 1990s, information technologies have been gradually integrated into all fields of our society and are playing a significant roll. Today, the emergence of wireless sensor networks(WSNs) may lead information technology to leap once again thus taking a step forward to narrow the distance of the human and physical world. As an important branch of WSNs, wireless multimedia sensor networks (WMSNs) combine the WSNs technology and multimedia technology. While WMSNs helps people to obtain abundant information such as images, audio and video, it also calls higher requirements correspondingly for research. The paper selects two significant aspects of WMSNs, acquisition and transmission of multimedia data, to research and implement. The work can make a contribution to making WMSNs more practical.

First of all, the prospects for development and research status at home and broad of WSNs and WMSNs are introduced. Secondly, the framework of system located in by the subjects in our study and establishment of devleopment platform are given. Then the research and implement of some key technologies of WMSNs are elaborated deeply, including the acquisition module and transmission module. The work on acquisition module consists of the modification of the kernel source, the realization of OV3640 control module, the optimization of some parameters of H.264 video encoder and the implement of acquisition of

audio data and video data. The work on transimission module includes the implements of the LT FEC, the statistics module of channel packet loss rate based on Gilbert model and the real-time feedback mechanism of optimal package length. At last, the detailed system tests on the results of the work above are done, and the test results indicate that the acquisition module and transmission module are able to meet the requirement of the paper.

KEY WORDS WMSNs, S3C6410, data acquisition, data transmission, LT encoding

目 录

第一章　绪论

1.1　课题背景

自 20 世纪 40 年代第一台数字式电子计算机诞生依赖，信息技术的每一次飞跃都会给人类的生活带来巨大的变化。第一次飞跃出现在 20 世纪 60 年代末，各行各业普遍引入了计算机系统，使得计算机硬件、软件技术蓬勃发展并出现了数据库、多媒体等产品和技术并在相当大的程度上改变了人类处理信息的方式。第二次飞跃出现在 1994 年前后，Internet 的诞生和兴起极大地丰富了人类的交流与沟通方式，其影响至今方兴未艾。第三次飞跃则以无线传感器网络（Wireless Sensor Networks, WSNs）的出现为代表，人类与物理世界的距离被进一步拉近，人类认识世界、改造世界的能力得到进一步加强。WSNs 的研究起源于 2000 年前后，它曾被美国媒体评为将对人类未来生活方式变革起到重要影响的前沿技术领域之一，其应用前景十分广阔，包括视频监控、交通路况监控、航空交通控制、机器人学、工业自动化等。

WSNs 中大量传感器节点被随机部署在监控区域，节点之间通过自组织方式构成网络。某个节点上监测到的数据被交付给相邻节点，通过一系列转发之后最终达到汇聚节点。汇聚节点可以处于 WSNs 的监控区域之内也可以处于 WSNs 的监控区域之外。汇聚节点收集到数据后，通过 Wifi、3G、卫星通信等方式将数据送至管理监控中心。终端用户通过管理监控中心发布命令、收集数据，从而实现对目标区域的监控和管理[1]。

目前已经有不少 WSNs 的应用实例，但大多数还处于应用的初级阶段，传输对象还仅局限于某些标量数据，包括光照强度、温湿度、电磁、噪声、土壤成分等等。随着业务类型的丰富和信息需求的提升，人们已不满足于从环境中获取简单的数据，而是希望可以得到图像、声音、视频等更为丰富的信息。在此背景之下，无线多媒体传感器网络（Wireless Multimedia Networks, WMSNs）应运而生。WMSNs 是在传统 WSNs 的基础上引入了图像、声音、视频等多媒体信息感知处理功能的一种新型网络[2]，它结合了 WSNs 技术和多媒体处理技术，能够实现比传统 WSNs 更高精度的监控，从而使用户更直观、更深入地了解观测对象。WMSNs 是 WSNs 的重要分支同时也是 WSNs 发展到高级阶段的网络形式。一方面，它继承了 WSNs 以数据为中心、分布式处理、动态拓扑、资源受限等特征；另一方面，以多媒体信息为采集、传输对象又对 WMSNs 各项技术的研究提出了更高的要求，包括硬件设计、能耗控制、QoS 等方面。二者的相同之处在于，节点资源有限、网络自主组织、拓扑动态变化、使用规模较大、数据多跳转发、应用相关性强。但另一方面，WMSNs 由于要处理多媒体数据，因此相比于 WSNs

具有几个明显的差异，如感知数据更丰富、处理任务更复杂、网络功能更强大等等。WMSNs 这些独有的特性，使其在获得广阔发展前景的同时对研究工作提出了更高的要求。

1.2 国内外研究现状

近年来各国密切关注对 WSNs 技术的研究，美国的“智慧地球”、日本的“u-Japan”、韩国的“IT839”、韩国三星集团的“U-City”计划、中国的“感知中国”战略等均已全面开展，研究领域包括 ZigBee 技术、超宽带技术、跨层优化、多媒体编解码、节点部署、网络覆盖、数据融合、服务质量保障等等。美国、欧盟、日本、韩国等地区/国家的 WSNs 研究的主要项目包括美国加州大学洛杉矶分校（UCLA）联合罗克韦尔研究中心负责的 WINS 项目[3]，该项目涉及传感器网络设计的各个方面，研究了 MEMS 传感器和接收器、信号处理结构、网络协议设计和监测理论的基本原理；美国麻省理工学院（MIT）负责的 μAMPS 项目设计了节能、自组织、可重构的 WSNs，设计低功耗的 μAMPS-I、μAMPS-II 传感节点，提出了以节能、可重构为主要目标的成组递阶网络通信协议 LEACH；美国加州大学伯克利（UCB）分校等 25 个研究机构负责的 SensIT 项目[4]计划共有 29 个研究子项目，实现“超视觉”战场监测；2003 年，ACM 专门组织国际视频监控与传感器网络研讨会（ACM International Workshop on Video-Surveillence & Sensor Networks）交流相关研究成果[5]。Feng 等人[6]设计了视频传感器节点 Panopts，视频流中图像帧分辨率可达 320×240，帧速率可达 18fps-20fps。Kulkarni 等人[7]组合了多种性能传感器模块和处理通信模块，搭建具有不同监测性能的视频传感节点。

我国在 WSNs 研究、应用及标准化等方面已达到世界一流水平。WSNs 技术已经成为我国信息领域少数位于世界前列的技术之一。由工信部资助，企业、高校、研究所单位共同参与的新一代宽带无线移动通信网国家重大专项，研制具有海量通信能力的新一代宽带无线通信接入系统、近距离无线互连系统与传感器网络，掌握关键技术，显著提高我国在国际主流技术标准所涉及的知识产权占有率，加大科技成果的商业应用并形成了超过 1000 亿元的产值规模。在 WMSNs 研究方面，南京邮电大学计算机学院的江苏省 WSNs 高技术研究重点实验室和 WSNs 研究中心从 2002 年开始就成立了专门的课题组，开始了 WMSNs 的研究与探索，在实现了多个版本的多媒体传感器节点和网关节点的同时还构建了一个高效的中间件软件开发平台。在此基础上，课题组开展了一系列理论学术方面的研究，包括协议设计、多媒体编码、数据采集、覆盖控制等方面，并积极尝试基于传感网的智能家居等应用系统的构建与测试。

1.3 课题研究意义及目标

在 WMSNs 众多关键技术的研究中有以下两个重要问题需要解决：一是如何实现有效、稳定、可靠的多媒体传感节点，为整个系统提供高质量的多媒体数据。WMSNs 的应用场景通常环境恶劣，对多媒体传感器节点的体积、功耗、所采集数据的质量、可靠性与稳定性等指标都有较高的要求。多媒体传感器节点的性能是 WMSNs 的重要衡量指标之一，其性能优劣直接影响着 WMSNs 的整体性能，可以说多媒体传感节点在 WMSNs 中占有举足轻重的低位。本文在参考前人研究成果的基础上，提出一种新的多媒体传感节点解决方案，即以 S3C6410 嵌入式平台为硬件基础并结合 OV3640、I^2C、H.264 和 ALSA 等多项技术实现的多媒体传感节点。S3C6410 具有体积小、耗电低、计算能力相对较强等特点，同时其内置的支持多种视频编解码的多媒体编码器 MFC 能在有效压缩视频数据量的同时保证实时性，为 WMSNs 提供有效、稳定、可靠的采集模块，具有一定研究意义；二是如何提供高质量的传输机制，缓解多媒体通信大数据量与 WMSNs 所采用无线信道的受限网络性能之间的矛盾。目前 WMSNs 带宽资源严重受限，信道抗干扰性差，使得支持实时可靠的大数据流媒体传输相当困难，如何改进传输机制以实现高质量的多媒体信息传输机制有待进一步研究。另一方面，多媒体数据具有普通数据所不具备的许多特点：第一，多媒体通信的海量数据在传输时对带宽的造成的压力远高于普通数据；第二，多媒体服务的特性使得其对传输网络的可靠性和实时性的要求很高，直接影响最终的用户体验；第三，节点采集到的多媒体数据具有大量的信息冗余不能直接传输，需要采用信源编码以降低冗余，而多媒体编码需要大量复杂计算，对硬件资源要求较高。因此，本文希望通过研究 WMSNs 的传输机制并实现稳定可靠的传输模块，在一定程度上缓解上述矛盾，使得 WMSNs 进一步实用化。

1.4 论文主要工作及内容组织

为实现本文的目标，论文开展的主要工作有：

（1）设计并实现 WMSNs 的采集模块。本文以 S3C6410 嵌入式平台为硬件基础，结合 OV3640 传感器驱动、I^2C 总线驱动、H.264 视频编码、ALSA 音频驱动等技术，设计并实现稳定可靠的采集模块。该模块能够采集高质量的视频、音频数据，同时还能够灵活控制 OV3640 传感器以及 H.264 的多项参数，采集模块的实现为本文的后续工作奠定了基础。

（2）研究 WMSNs 传输机制并实现 WMSNs 的传输模块。本文提出将 LT 编码引入 WMSNs。LT 编码是由 Michael Luby 提出的一种前向信道编码技术，码率动态可变。其以一定数据冗余为代价，在收到任意一组稍多于原始数据分组的编码分组后，就能正确恢复出所有的原始数据分组[8]。LT 编码的引入能有效

提高 WMSNs 的抗干扰性，确保多媒体通信质量。同时本文还设计了基于 LT 的动态最优分组长度反馈机制，该机制能在确保 LT 编码质量的同时有效降低 LT 编码带来的数据冗余。

全文结构安排如下：

第一章 绪论。该部分综合介绍 WSNs 与 WMSNs 的发展状况和国内外研究现状，阐述本文研究意义和研究目标，并说明本论文的主要工作内容。

第二章 系统总体框架与开发平台的搭建。该部分给出 WMSNs 的总体框架以说明本文所涉及部分在系统中的功能，同时介绍了本文所采用的开发平台的搭建过程。

第三章 采集模块的设计与实现。该部分给出采集模块的详细设计、涉及到的相关技术以及各部分的具体实现过程。

第四章 传输模块的设计与实现。该部分给出传输模块的详细设计、涉及到的相关技术以及各部分的具体实现过程。

第五章 实验设计与测试。该部分对采集模块和传输模块进行测试，包括实验环境的搭建、实验的过程以及实验的结果。

第六章 总结与展望。对整个论文工作进行总结，说明本文工作的几项待改进之处并给出改进建议。

第二章　系统总体框架及开发平台的搭建

2.1　系统总体框架

图 2-1 所示是本文所应用的 WMSNs 的总体框架，整个系统包含三部分：采集节点、中继节点和汇聚（Sink）节点。

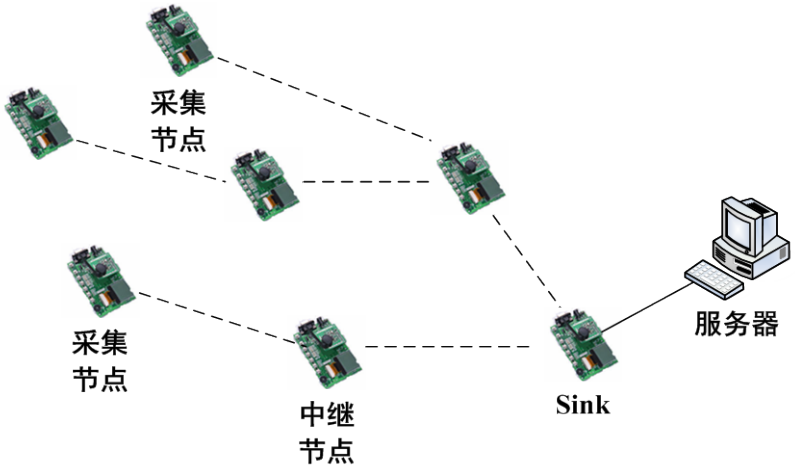

图 2-1　系统总体框架

整个系统的运作过程为：采集节点借助搭载于其上的视频传感器和音频传感器感知外界信息并捕获数据，并将这些数据进行信源编码（如 H.264 视频编码）和信道编码（如 LT 前向信道编码），然后发送至相邻的中继节点。中继节点的主要职责是自动寻路并对数据包进行转发。Sink 节点汇聚最终接收到的数据并针对不同场景做不同的应用处理，如车牌识别、人脸识别、人源识别等，最后将处理结果交由服务器端显示。本文所涉及内容主要为采集节点中的数据采集和数据传输部分，此处称之为采集模块和传输模块。

2.2　开发平台的搭建

2.2.1　VMware 虚拟机和 Ubuntu 的安装配置

为了顺利完成后续开发和调试工作，本文采用如下两种解决方案：PC 端开发+交叉编译+直接烧录，或 PC 端开发+交叉编译+NFS 挂载，因此首先需要搭建 PC 端的开发环境，本文采用 VMware 7.0.0+Ubuntu 10.04 的方案。之所以采用虚拟机而不直接在 PC 上安装 Ubuntu，是考虑到后续使用的串口调试工具 PuTTy 和内核/文件系统烧录工具 DNW 只能在 Windows 平台上运行，因此使用虚拟机可以降低频繁在两个系统间的切换时造成的时间开销，提高开发效率。VMware 7.0.0 和 Ubuntu 10.04 的安装包可以方便地从互联网上获得，其安装过程不再赘述，此处仅介绍安装完成后需要进行的几项主要配置。

（1）安装 VMware Tools，该工具可以使用户方便地在物理主机和虚拟机之间拖曳传输文件，极大地提高工作效率。具体方法是在 VMware 的菜单栏里选择 VM-Install VMware Tools。

（2）设置 VMware 网络适配器。在 VMware 最下方状态栏中选择 Network Adaptor-Settings，可看到如图 2-2 所示的设置界面中有四种网络连接方式 Bridged、NAT、HLinuxt-only 以及 Custom，这里一般选择 Bridged 或 NAT。二者的区别在于：Bridged 直接连向物理网络，因此虚拟机系统会获得唯一的因特网 IP 地址；NAT 采用了网络地址转换技术，实际上与物理主机共享同一个因特网 IP。一般来说，在 IP 受限的情况下应采用 NAT 方式。

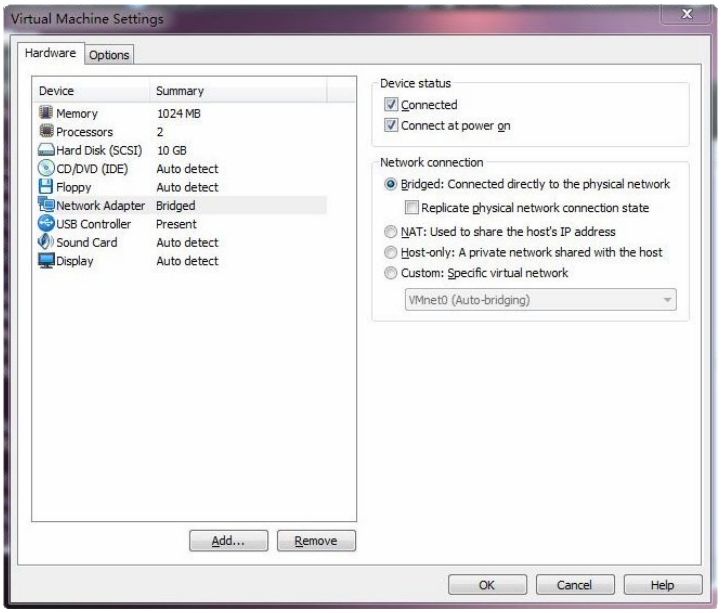

图 2-2 VMware 网络适配器设置

（3）设置 Ubuntu 网络配置。在本文的实际调试和测试过程中，所有传感器节点均采用了 10/8 的内网地址进行通信，因此为使 Ubuntu 能正常与节点通信，要对 Ubuntu 添加相应的内网 IP，即使用如下命令（假设 Ubuntu 的内网 IP 为 10.0.0.87）

```
sudo ifconfig eth0:0 10.0.0.87 netmask 255.255.255.0
```

2.2.2 交叉编译器的安装配置

本文的开发工作主要在 PC 端完成，由此带来一个问题，即 PC 端采用 x86 架构而 S3C6410 嵌入式平台采用的是 ARM 架构，因此在 Ubuntu 下直接编译得到的 x86 可执行文件无法在 S3C6410 嵌入式平台中执行，本文采用交叉编译工具来解决该问题。交叉编译器可以在某种架构平台（如 x86）上编译出可运行于其他架构平台（如 ARM）的可执行文件。本文采用由 ARM 公司推出的新一代

ARM 交叉编译器 EABI。EABI 使用了新的 glibc 库 2.8，并在编译器中预先安装好各种需要用到的库文件，如编译 qtopia 时需要用到的 jpeg、zlib、libts、libuuid 等等，使得在编译时不需要额外安装任何第三方函数库，带来极大方便。安装配置 EABI 的步骤如下：

（1）解压，使用如下命令得到目录 usr

```
tar –xvr arm-linux-gcc-4.3.2.tgz
```

（2）将步骤（1）所得 usr/local/中的 arm 文件夹复制至/usr/local，使用命令

```
cd usr/local
sudo cp –r arm /usr/local
```

（3）编辑配置文件/etc/profile，添加如下一行并保存

```
PATH=/usr/local/arm/4.3.2/bin:$PATH
```

（4）使环境变量设置立即生效，使用命令

```
source /etc/profile
```

（5）测试配置是否成功，使用如下命令，若显示如图 2-3 所示的结果则表示 EABI 配置成功

```
arm-none-linux-gnueabi-gcc –v
```

```
Using built-in specs.
Target: arm-none-linux-gnueabi
Configured with: /scratch/julian/lite-respin/linux/src/gcc-4.3/configure --build
=i686-pc-linux-gnu --host=i686-pc-linux-gnu --target=arm-none-linux-gnueabi --en
able-threads --disable-libmudflap --disable-libssp --disable-libstdcxx-pch --wit
h-gnu-as --with-gnu-ld --enable-languages=c,c++ --enable-shared --enable-symvers
=gnu --enable-__cxa_atexit --with-pkgversion='Sourcery G++ Lite 2008q3-72' --wit
h-bugurl=https://support.codesourcery.com/GNUToolchain/ --disable-nls --prefix=/
opt/codesourcery --with-sysroot=/opt/codesourcery/arm-none-linux-gnueabi/libc --
with-build-sysroot=/scratch/julian/lite-respin/linux/install/arm-none-linux-gnue
abi/libc --with-gmp=/scratch/julian/lite-respin/linux/obj/host-libs-2008q3-72-ar
m-none-linux-gnueabi-i686-pc-linux-gnu/usr --with-mpfr=/scratch/julian/lite-resp
in/linux/obj/host-libs-2008q3-72-arm-none-linux-gnueabi-i686-pc-linux-gnu/usr --
disable-libgomp --enable-poison-system-directories --with-build-time-tools=/scra
tch/julian/lite-respin/linux/install/arm-none-linux-gnueabi/bin --with-build-tim
e-tools=/scratch/julian/lite-respin/linux/install/arm-none-linux-gnueabi/bin
Thread model: posix
gcc version 4.3.2 (Sourcery G++ Lite 2008q3-72)
```

图 2-3　EABI 配置成功时显示信息

2.2.3　NFS 服务器的安装配置

本文 2.2.1 节中提到，本文采用两种解决方案进行开发、调试及测试：一是 PC 端开发+交叉编译+直接烧录，二是 PC 端开发+交叉编译+NFS 挂载。直接烧录是将程序和内核一同编译并下载到目标板，这种模式存在一个问题即每次调试都需要将程序烧入 nandflash，还经常需要根据不同的需求重新烧写文件系统和

内核，这样的反复操作不仅使得开发效率低下，还会在一定程度上对 nandflash 产生损耗。NFS 是 Network File System 的缩写，最早由 Sun 公司提出。NFS 允许不同的客户端及服务端通过一组 RPC 分享相同的文件系统。换而言之，NFS 服务器可以让主机将网络远端的 NFS 服务器共享的目录挂载到本地端的主机当中，在本地端的主机看来，远端主机的目录就好像自己本地磁盘的一个分区一样，使用起来更加方便。因此，使用 NFS 挂载能够避免对 nandflash 的反复烧写，降低对 nandflash 的损耗并提高开发效率。

NFS 服务器的安装配置方法如下：

（1）安装 NFS Server

```
sudo apt-get install nfs-kernel-server nfs-common portmap
```

（2）配置 portmap，编辑/etc/default/portmap，将最后一行注释起来

```
###OPTIONS="……"
```

（3）配置挂载目录和权限，使用命令

```
sudo chmod 777 /etc/exports
```

（4）编辑/etc/exports，添加如下一行。其中/nfsboot 是 NFS 的共享目录，* 表示任何 IP 都可以共享该目录，rw 表示读写权限，sync 表示对此目录内容同步更新。

```
/nfsboot *(rw,sync)
```

（5）更新 exports 文件，使用命令

```
sudo exportfs –r
```

（6）重启 NFS 服务，使用命令

```
sudo /etc/init.d/nfs-kernel-server restart
```

（7）测试是否安装配置成功（假设 Ubuntu 的 IP 为 10.0.0.87），使用命令

```
sudo mount 10.0.0.87:/nfsboot /mnt   //挂载
df   //查看是否挂载成功
umount /mnt   //解挂载
```

2.2.4 PuTTy 的配置

PuTTy 是一个 Telnet、SSH、rlogin、纯 TCP 以及串行接口连接软件。PuTTy 包括了 IPv6 连接；可以控制 SSH 连接时加密协定的种类（目前有 3DES、AES、Blowfish、DES 及 RC4）；CL 版本的 SCP 及 SFTP Client，分别叫做 pscp 与 psftp；自带 SSH Forwarding 的功能，包括 X11 Forwarding；完全模拟 xterm、VT102 及 ECMA-48 终端机的能力；支持公钥认证。本文使用 PuTTy 对 S3C6410 进行串口调试，其配置参数如图 2-4 所示。本文采用串口进行 S3C6410 的调试，因此连接

类型选择“Serial”；端口号应与设备管理器中识别到的串口端口号一致，本文为COM1；由于是本地有线连接，因此将数据速率设为115200，以达到较好的用户体验。

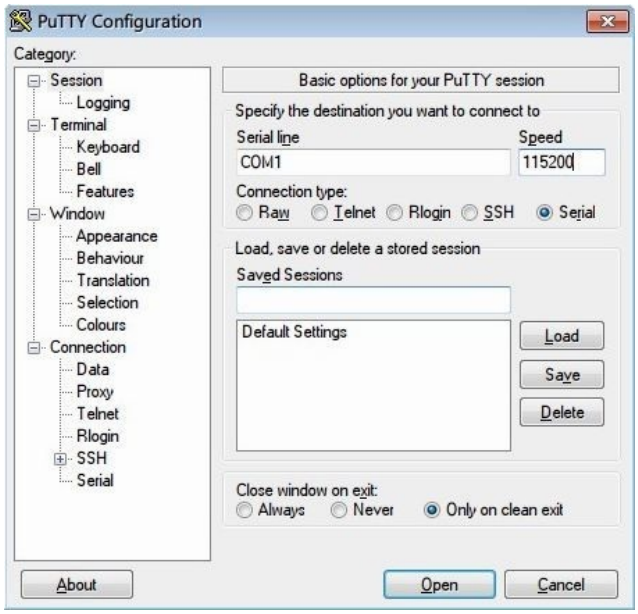

图 2-4　PuTTy 配置

第三章　采集模块的设计与实现

3.1　采集模块的总体设计

本文所实现的采集模块的主要功能包括采集视频数据和音频数据并对视频数据进行视频编码压缩，之后将音频数据和压缩后的视频数据送至传输模块。此外，采集模块还负责通过 I^2C 总线控制 OV3640 视频传感器，以适应不同场景需要。采集模块的总体设计如图 3-1 所示。

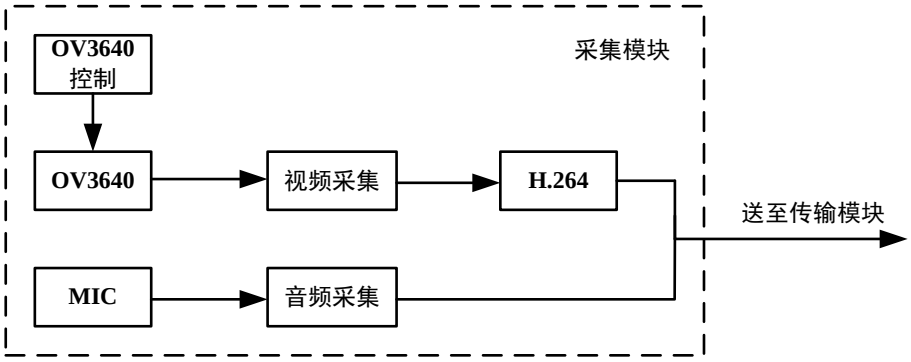

图 3-1　采集模块总体设计

如图 3-1 所示，实现采集模块需要完成的工作有以下几部分：

（1）修改 Linux 内核源码。本文要求通过 OV3640 视频传感器采集到不同分辨率的高质量视频数据，帧速率在 20-30fps 之间，视频分辨率为 480*272 和 640*480 两种，原始视频数据格式为 YUV。但由于 S3C6410 嵌入式开发平台自带内核的限制，所采集视频数据难以达到本文的指标要求，因此需要对 Linux 内核源码进行一定的修改。

（2）通过 I^2C 总线实现对 OV3640 视频传感器的各种控制，包括数码变焦、图像的镜像和翻转、曝光控制以及增益控制等等。工作焦点在于如何实现有效的 API 对 OV3640 视频传感器的内置寄存器进行读写并利用该 API 设计实现 OV3640 实时控制程序。

（3）调用 H.264 视频编码器并优化其部分参数。视频编码并非多媒体数据采集的必要环节，但由于多媒体通信中存在着海量数据，对无线信道有限的带宽造成了很大压力，因此采用信源编码即 H.264 视频编码能在确保图像质量有限失真的同时有效地压缩数据量，为之后的数据传输提供便利。本部分的工作焦点在于如何实现调用 H.264 编码器以及优化 H.264 参数使其能够适应本文所应用的网络环境。

（4）实现对视频数据的采集。本文要求通过 OV3640 视频传感器采集高质量视频数据，要求其分辨率达到 480×272 或 640×480，帧速率达到 20-30fps。

本部分的工作焦点在于如何实现视频采集程序。

（5）ALSA 音频驱动的移植。完成此项工作是实现音频采集的前提条件，因为在 S3C6410 嵌入式平台中默认音频驱动为 OSS，只有成功移植了 ALSA 音频驱动才能在后续工作中利用其提供的 API 设计实现音频采集程序。本部分的工作焦点在于如何有效配置交叉编译 ALSA 音频驱动源码的参数，以及如何将编译好的 ALSA 驱动部署至 S3C6410 嵌入式平台。

（6）实现对音频数据的采集。本文要求通过 S3C6410 嵌入式平台内置的 MIC 采集高质量音频数据。本部分的工作焦点在于如何利用 ALSA 音频驱动提供的 API 设计实现音频采集程序，并灵活地调整某些参数，如采样率、采样位数、声道数等等。

3.2 S3C6410 嵌入式开发平台

本文的采集模块是以 S3C6410 嵌入式开发平台为硬件基础的，因此下面首先对该平台进行简单的介绍。

3.2.1 嵌入式系统

嵌入式系统是一种特定的计算机系统，用于在更大的系统中实现特定的功能。换言之，嵌入式系统通常作为嵌入在电子设备中的一部分而存在。相比而言，包括 PC 机在内的通用计算机的设计更加灵活，更能够满足用户的各项需求。而嵌入式系统的优势在于它是被用来完成特定的任务，因此设计者可对其进行优化从而降低嵌入式系统的尺寸和成本，同时增强其可靠性和性能。嵌入式系统的应用实例包括自动柜员机 ATM、移动电话和电信交换机、办公设备如打印机、复印机、传真机等。嵌入式系统作为计算机结构中的重要分支与普通计算机既有异同点也有相同点，相同点在于它在硬件上的组成与标准的计算机类似，其中最主要的部分也是微处理器。与标准的计算机结构相同，嵌入式系统中也包含了中央处理器、内存、输入/输出设备，只不过在嵌入式系统里，这些单元以比较特殊的形式存在，如家用电器采用触控面板取代键盘作为标准输入[9]。嵌入式系统一般由三个主要部分组成：（1）硬件。图 3-2 给出了嵌入式系统硬件中的组成单元，其中处理器是系统的运算核心，存储器用来保存可执行代码和中间结果，输入输出设备完成与系统外部的信息交换，其他部分辅助系统完成功能；（2）应用软件，应用软件是完成系统功能的主要软件，可以由单独的一个任务来实现，也可以由多个并行的任务来实现；（3）实时操作系统，用来管理应用软件，并提供一种机制，使得处理器分时地执行各个任务并完成一定的时限要求。

嵌入式系统的软件设计受三个条件的限制：（1）系统存储器容量；（2）处理

器速度；（3）当以等待事件、运行、停止和唤醒的周期连续运行系统时，对功耗的限制。

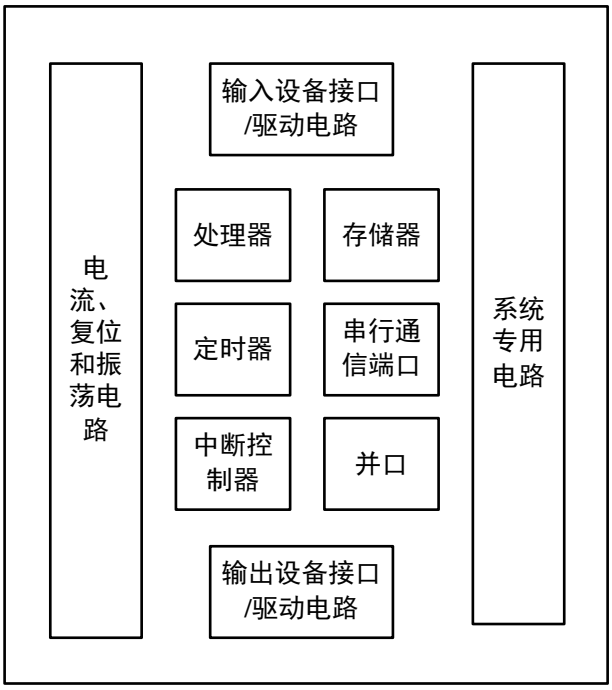

图 3-2　嵌入式硬件系统组成

3.2.2　基于 ARM 架构的嵌入式系统

嵌入式微处理器类型主要有 386EX、Power PC、68000、MIPS、ARM 系列等。ARM 微处理器是指按照 ARM 公司提出的 RISC 处理器设计开发架构所生产的微处理器系列。ARM 架构出现于上世纪 80 年代，现如今按其标准设计生成的许多 32 位指令集架构产品取得了非常广泛的应用。由于采用了 RISC 指令集，实现核心 ARM 处理器仅需要 35000 个晶体管，而实现传统处理器芯片需要数以百万计的晶体管。因此 ARM 处理器的功耗要远低于传统处理器，从而更适合应用于小型电子设备。截止 2005 年，世界上每年销售的约 10 亿部移动电话中有 98%采用了 ARM 微处理器。截止 2009 年，ARM 微处理器占所有 32 位 RISC 嵌入式处理器总量的 90%且被广泛应用于消费电子设备，如移动电话、数字媒体音乐播放器、计算器、硬盘驱动器及路由器等。为了保证设计的简洁和高效，原始的 ARM 架构实现并未采用微内核而是转而采用了硬连接形式，这点类似于较为简单的 8 位 6502 处理器。

ARM 架构包括以下 RISC 特性：

（1）加载/存储架构；

（2）不支持未对齐内存访问（当前的 ARMv6 版本的内核支持该特性，但在加载/存储多个字指令时例外）；

（3）固定 32 位指令宽度，更适合于解码和流水线，且编码开销更低；

（4）统一采用单时钟周期执行。

同时为了弥补简单化设计带来性能上的不足，ARM 架构采用了以下一些额外的设计：

（1）对大多数指令采用条件化执行方案以减少分支开销；

（2）只在必要时改变算术指令的条件代码；

（3）绝大多数算术指令和地址计算采用了 32 位柱式位移器以避免性能的降低；

（4）强大的索引寻址模式；

（5）采用链式寄存器从而使得函数调用更加迅速。

3.2.3 S3C6410 嵌入式开发平台

本文所实现的传感器节点采用了 S3C6410 嵌入式平台，简称 S3C6410。该平台搭载了由三星公司推出的基于 ARM 架构的 S3C6410 微处理器。除了微处理器，S3C6410 嵌入式平台还包含其他若干组成部分，如图 3-3 所示，S3C6410 嵌入式平台由核心板和底板（外设板）两部分组成。核心板的尺寸仅相当于一个 48mm*67mm 的方块的大小，其中集成了基于 ARM11 的 S3C6410 处理器、128MB 的 DDR 内存以及 1GB 的 NANDFLASH，同时预留了 256K 的 NORFLASH。底板提供了丰富的外设接口，如 RS-232 串口、USB 接口（一个 host，一个 device）、10M/100M 自适应以太网接口、SD 卡接口、MFC 接口以及 TFT LCD 接口等等。

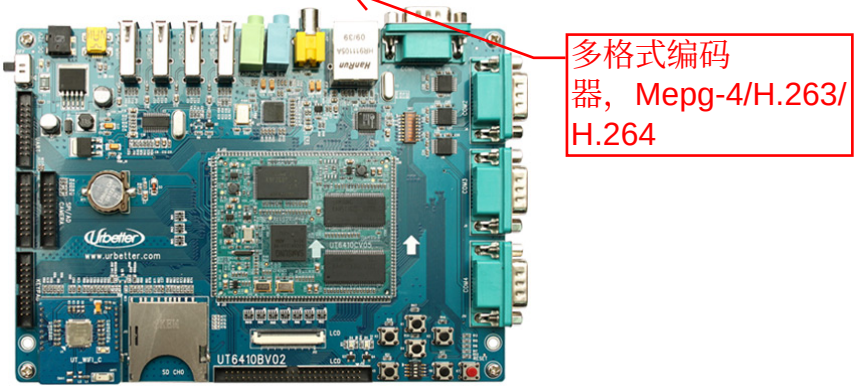

图 3-3　S3C6410 嵌入式开发平台

图 3-4 给出了 S3C6410 硬件结构。同时系统提供了一系列软件资源包括 s3c-u-boot-1.1.6 引导程序、s3c-Linux-2.6.28.4 内核、ubifs/yaffs2/cramf/fat32 文件系统、qtopia-2.2.0 和 QtE-4.5.2 图形界面以及一系列设备驱动程序，用户可利用这些资源在系统平台上进行自主软件开发。

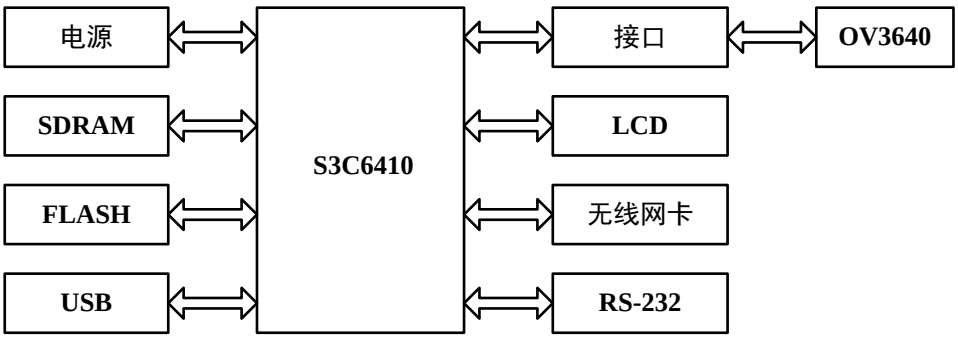

图 3-4　S3C6410 硬件结构

3.3　Linux 内核源码的修改

3.3.1　Linux 内核

本文采用了 S3C6410 嵌入式开发平台为硬件基础，但存在一个问题，即其默认内核限制了视频采集的分辨率和帧速率，导致采集到的视频数据不能达到本文的要求，因此需要对内核源码进行一定修改，再编译、烧录至 S3C6410。内核是 Linux 中的基本结构单元，它可以定义为 Linux 中一个运行于管态的安全单元，而 Linux 的其他部分和应用软件都运行于用户态。Linux 内核功能包括：进程管理、存储器管理、设备管理以及文件系统的组织和实现[10]。

（1）进程管理。初始的进程是一个在处理器重启时需要被操作系统执行存储器指令，然后调用 Linux 的进程。处理器开始执行随后创建的所有进程。创建指的是为创建的进程定义地址空间（存储器块），并为进程定义资源。进程可以以继承的方式创建。进程管理程序在创建进程时分配 PCB，并通过 PCB 对进程进行管理。其他的 Linux 单元可以在必要的时候向操作系统请求查询进程的 PCB。进程管理程序能够进行进程的创建、激活、运行、阻塞、再运行、释放以及删除。进程管理程序响应进程对资源或者 Linux 服务的请求，然后同意这一请求，让进程共享资源。正在运行的进程可以通过消息和系统调用两种方法发出请求。消息的原理是运行于用户态的进程产生并发出一个消息，以便被请求资源能够使用或者运行一个 Linux 服务函数。系统调用即 Linux 中的函数调用，首先发送一条指令使处理器发生自陷并切换到管态，这时 Linux 可以像执行库函数一样执行函数。一旦完成了调用函数的指令，处理器就从管态切换到用户态，并让正在调用的进程继续执行。

（2）存储器管理。当进程创建的时候，存储器管理程序通过映射进程地址空间来为进程分配存储器地址（存储器块）。Linux 的存储器管理程序必须是健壮的、完善的、灵活的且受到良好保护的。除了存储器泄漏和堆栈上溢以外，不能有任何的错误发生。存储器泄漏指的是试图向没有分配进程或数据结构的存储

块中写入数据。堆栈上溢指的是在没有提供额外堆栈空间的情况下，超出分配给堆栈的存储块。存储器管理的内容包括：管理进程对存储器地址空间的使用，限制给定存储空间共享的特殊机制，通过使用存储器层次结构，即缓存、基本的或者扩展的二级存储体和光存储体的层次结构来优化访存周期。存储器管理程序为进程分配存储空间，并使用适当的保护机制来进行管理。存储器分配可以是静态的，也可以是动态的，这样设计的目的是为了优化存储器的需求，在不增加存储器读取开销的基础上提高存储器利用率。

（3）设备管理。操作系统中存在大量设备驱动程序的 ISR，每一个设备或者设备功能都有一个单独的与其硬件对应的驱动程序，设备管理程序是管理这些驱动程序的软件。Linux 操作系统的设备管理程序提供并执行用于管理设备和它们的驱动程序 ISR 的模块。设备管理程序的主要功能包括设备检测和添加，设备删除，设备分配和注册，删除和注销，设备共享，设备缓冲区管理等。

（4）文件系统的组织和实现。文件是磁盘、光盘或者系统存储器上的一个命名实体。文件包含数据、字符和文本或者它们的组合。如同每个进程都包含一个 PCB（进程控制块）一样，每个文件系统都有包含一个称为文件描述符的数据结构。直到文件关闭前，文件描述符都始终可用。文件管理程序的主要功能包括创建文件、打开文件、读取文件、写入文件、查找记录以及关闭文件。

3.3.2 Linux 内核源码的修改

为使采集到的视频数据的分辨率能达到 640×480，帧速率能达到 20-30fps，需要对内核源码进行一定修改，步骤如下：

（1）编辑源码/drivers/media/video/Samsung/fimc/ov3640.h，修改其中二维数组 ov3640_setting_30fps_VGA_640_480 的内容，写入如下几行。其中第一行可以解除原内核对视频帧速率的限制，第二行和第三行使得 OV3640 能够支持 640×480 分辨率。

```
{0x30,0x12, 0x80},
{0x30,0x88, 0x03},{0x30,0x89, 0x20},{0x30,0x8a, 0x02},{0x30,0x8b, 0x58},
{0x30,0x8d, 0x04},{0x30,0x86, 0x03},{0x30,0x86, 0x00},{0x36,0x00, 0xc4}
```

（2）编辑源码/drivers/media/video/Samsung/fimc/camera.c，将如下代码中的 ov3640_setting_15fps_VGA_640_480 改为 ov3640_setting_30fps_VGA_640_480。

```
for(int i = 0; i < sizeof(ov3640_setting_15fps_VGA_640_480)/
sizeof(ov3640_setting_15_fps_VGA_640_480[0]; i++){
    s3c_fimc_i2c_write(client, ov3640_setting_15fps_VGA_640_480[i],
    sizeof(ov3640_setting_15fps_VGA_640_480[i]); }
```

（3）编译源码并烧录至 S3C6410。

3.4 OV3640 控制模块

3.4.1 OV3640 视频传感器

为使多媒体传感节点能正常采集视频数据，本文采用 OmniVision 公司推出的 OV3640 视频传感器进行视频数据的采集并利用 I^2C 对 OV3640 进行多种控制。

刀片摄像头

OV3640 视频传感器是一款功能丰富的高性能 300 万像素 CMLinux 视频传感器，采用 1/4 英寸的光学规格。它基于 OmniVision 的 1.75 μ m OminiPixel3 架构设计并具有 ULSH(Ultra Low Stack Height)特性，从而获得了出色的低光感灵敏度（500mV/lux-sec）并显著改善了动态信噪范围（65db）[11]。OV3640 视频传感器的图像阵列包含 1568 行和 2072 列，共 3248896 像素。OV3640 的色彩过滤器采用了 Bayer 排列方式且其像素按照 BG/GR 阵列交替的方式进行分布。在所有的 3248896 像素中，有 3145728（2048×1536）个像素是活动的且能够被输出，剩余像素用于校正和内插。OV3640 视频传感器支持包括 RGB 和 YUV 在内的多种视频格式的输出，同时支持包括 QXGA、XGA、VGA 在内的多种视频尺寸。OV3640 视频传感器的视频采集速率最高可达 30 帧/秒，能够满足大部分应用场景。

本文选取 OV3640 视频传感器的另一个原因在于可通过 I^2C 总线对 OV3640 视频传感器进行多种控制，包括数码变焦、图像翻转、曝光控制以及增益控制等等。本文利用 I^2C 总线将 OV3640 视频传感器连接到 S3C6410 嵌入式平台上，OV3640 视频传感器将绝大部分控制信息存放于其内部的寄存器中，因此本文对 OV3640 视频传感器的控制主要方式是读写其相应的寄存器值。

3.4.2 I^2C 总线

I^2C（Inter－Integrated Circuit）总线是由 Philips 公司开发的两线式串行总线，用于将低速率的外围设备连接至主板、嵌入式系统、移动电话或其他电子设备中。I^2C 包含两条双向漏极开路线，即串行数据线（SDA）和串行时钟线（SDL），在目前众多总线标准中 I^2C 使用的线路数最少，这也是 I^2C 的优点之一。I^2C 的系统结构如图 3-5 所示，可以看到，每个 I^2C 设备都通过串行数据线和串行时钟线与 I^2C 总线控制器相连。I^2C 总线实现了包括自动寻址、高低速设备速率匹配和全双工通信等功能[12]。

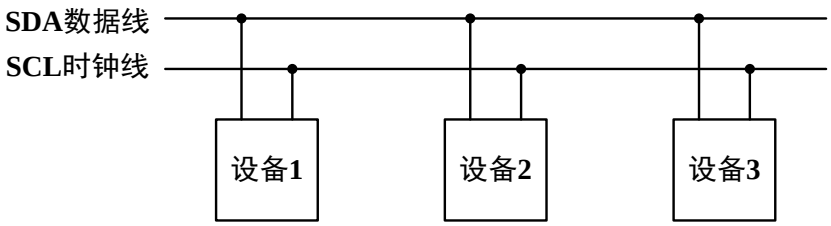

图 3-5　I^2C 系统结构图

在 I^2C 总线机制下所有设备都具有唯一的标识即设备地址。I^2C 支持数据的双向传输，因此系统中的多个设备均可作为数据源进行数据发送。但由此产生一个问题即在某一时刻只能由一个设备独占总线资源，对此 I^2C 总线采用仲裁机制以避免冲突。

3.4.3　OV3640 的主要寄存器

在实现了 I^2C 读写 API 后，就可以利用该 API 开发 OV3640 控制模块。此项工作实际上就是找到控制 OV3640 视频传感器的相应寄存器并对其进行读写操作以实现控制效果，本文用到的寄存器如表 3-1 所示。

表 3-1　OV3640 主要寄存器及相应功能

地址	功能描述
0x307C[0]	垂直翻转
0x3090[3]	水平翻转
{0x3020, 0x3021} {0x3022, 0x3023} {0x3024, 0x3025} {0x3026, 0x3027}	裁剪
0x3013[0]	自动曝光控制自动/手动选择
0x3013[2]	自动增益控制自动/手动选择
0x335F 0x3360 0x3361	数码变焦

这里对表 3-1 给出的寄存器的功能做一简单介绍：

（1）垂直翻转（mirror）和水平翻转（flip）。OV3640 提供了 flip 和 mirror 输出模式，效果如图 3-6 所示，将寄存器 0x307C[0]的值设为 1 则表示开启 flip 模式，为 0 则表示关闭；将寄存器 0x3090[3]的值设为 1 则开启 mirror 模式，为

0 则关闭。在 mirror 模式下，由于 Bayer 排序方式由 BGBG 变为 GBGB，OV3640 会将输出序列延后一个像素（通过将寄存器 0x397C[1]的值设为 1）。而在 flip 模式中，OV3640 不需要做额外的设置，因为 ISP 时钟会自动检测像素是否在红色列或蓝色列中，并做相应的校正。

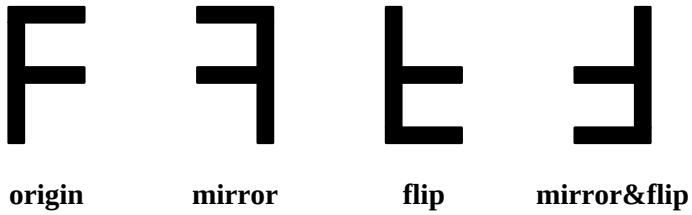

图 3-6　mirror 和 flip 效果

（2）裁剪。图像裁剪区域由四个参数定义，即水平起始点 HS、水平宽度 HW、垂直起始点 VS 以及垂直宽度 VW。通过设置合适的参数即设置 0x3020-0x3027 这几个寄存器的值，传感器阵列的任何部分均可被裁剪出来并做呈现。一般来说，裁剪是通过简单地屏蔽裁剪窗口之外的像素来实现的，因此裁剪并不会影响原始图像，也不会与 mirror 和 flip 冲突。

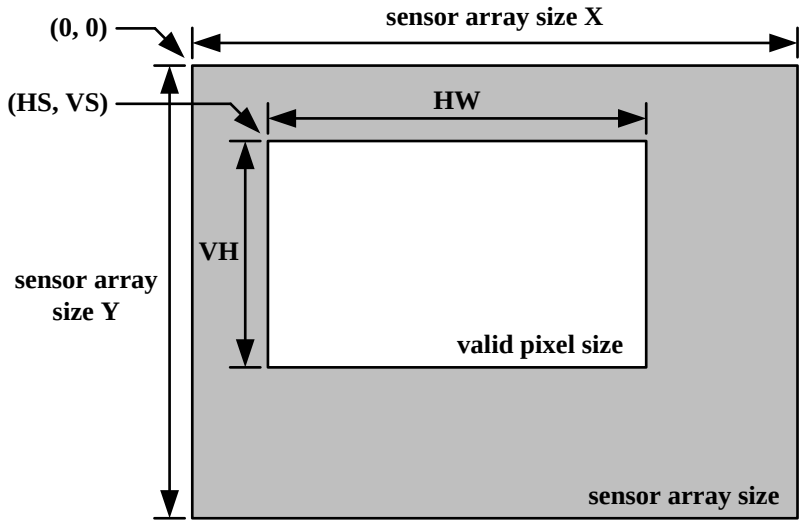

图 3-7　裁剪效果

（3）自动曝光控制（AEC）。 AEC 的作用是计算下一帧积分的时长并将该信息发送至计时控制块。AEC 能够基于前一帧的统计信息来决定是否对积分时长进行增加、减少、快速增加、快速减少或保持不变。在极高亮度的情况下，使用 LAEC 模式可将积分时长降低至一行以下。而在极低亮度的情况下，使用 VAEC 模式可将积分时间提高至一帧以上。另一方面，为了避免由外部光照强度周期性变化而导致图像闪烁，可将积分时间步长设为外部光源光照强度变化周期的整数倍。将寄存器 0x3013[0]的值设为 1 表示开启 AEC，为 0 则表示关闭 AEC。

（4）自动增益控制（AGC）。提高增益主要是通过在曝光时间的上下界范围内放大信号和噪声，而不是去延长积分时间。因此通常在 AEC 值达到上界后才开始启用 AGC 功能。但是如果相邻 AEC 步长过大（大于 1/16）则此时需要引入 AGC，否则积分时间会持续在两个相邻步长间切换，从而导致图像闪烁。将寄存器 0x3013[2]的值设为 1 开启 AGC，为 0 则关闭 AGC。

（5）数码变焦。数码变焦按照新图像的宽度和高度使用若干个像素值来生成单一像素值。有时需要将某些像素值均分并应用于两个或者更多个相邻像素。通过设置 0x335F、0x3360 和 0x3361 三个寄存器的值可实现数码变焦。

3.4.4 I^2C 读写 API 的封装

对 OV3640 的控制实际上是对 OV3640 内置的多个寄存器进行读写操作。本文通过 I^2C 总线对 OV3640 相应的内置寄存器进行读写，因此首先需要实现封装好的 I^2C 读写 API，以便后续实现的 OV3640 控制模块进行调用。实现 I^2C 读写 API 的过程中，主要用到了以下两个结构体：

```
struct I2cMsg{
    unsigned short Addr;   //设备地址,这里恒为 OV3640_ADDR>>1   (OV3640 = 0x87)
    unsigned short Flags; //0 代表写，1 代表读
    unsigned short Len;   //buf 的长度
    unsigned char * Buf;   //信息数组
};
struct I2cRdwrIoctlData{
    struct I2cMsg * Msgs;
    int Nmsgs;   //Nmsgs 这个数量决定了有多少开始信号，对于“单开始时序”，取 1
};
```

在结构体 struct I2cMsg 中，成员变量 Addr 标识了设备地址，因为本文只通过 I^2C 操作 OV3640，因此这里 Addr 的值恒为 0x87>>1；成员变量 Flags 标识了本次行为是读操作还是写操作，其中 1 表示读操作，0 表示写操作；成员变量 Buf 是一个 char 数组，存储了需要写入寄存器或从寄存器读出的数据信息；成员变量 Len 标识了 Buf 数组的大小。因此可以看到一个 struct I2cMsg 结构体变量容纳了一份写入寄存器或从寄存器读出的信息。在结构体 struct I2cRdwrIoctlData 中，成员变量 Msgs 是一个指向 struct I2cMsg 类型的指针，其为以 struct I2cRdwrIoctlData 类型为结点的链表的表头指针；成员变量 Nmsgs 标识该链表的长度。在 I^2C 的读写操作中，读操作需要两个 Msgs 信号，故其链表长度为 2，而写操作只需要一个 Msgs 信号。I^2C 的读操作和写操作的封装比较类似，由于

篇幅限制这里只给出 I^2C 读操作的实现伪码：

```
    void I2cRead(int I2cHandle, unsigned char AddrHigh, unsigned char AddrLow, char *
Buffer){
        struct I2cRdwrIoctlData E2PromData;
        E2PromData.Nmsgs = 2;
        E2PromData.Msgs =
        (strct I2cMsg *)malloc(E2PromData.Nmsgs*sizeof(struct I2cMsg));
        //读操作包含两个 Msgs 信号，Msgs[0]存放目标寄存器地址，Msgs[1]存放读取内容
        //对于 Msgs[0]需要写入寄存器，因此 Flags 值为 0
        E2PromData.Msgs[0].Len = 2;
        E2PromData.Msgs[0].Addr = OV3640_ADDR>>1;
        E2PromData.Msgs[0].Flags = 0;
        E2PromData.Msgs[0].Buf =
        (unsigned char *)malloc(E2PromData.Msgs[0].Len*sizeof(unsigned char));
        //对于 Msgs[1]需要从寄存器读出，因此 Flags 值为 1
        E2PromData.Msgs[0].Buf[0] = AddrHight;
        E2PromData.Msgs[0].Buf[1] = AddrLow;
        E2PromData.Msgs[1].Len = 1;
        E2PromData.Msgs[1].Addr = OV3640_ADDR>>1;
        E2PromData.Msgs[1].Flags = 1;
        E2PromData.Msgs[1].Buf = (unsigned char *)malloc(sizeof(unsigned char));
        E2PromData.Msgs[1].Buf[0] = 0;   //初始化读缓冲
        *Buffer = E2PromData.Msgs[1].Buf[0];   //将读出的内容存放至 Buffer
    }
```

函数参数中，I2cHandle 标识设备的句柄号，在本文里就是 OV3640 的句柄号；AddrHigh 和 AddrLow 分别标识要操作寄存器地址的高位和低位；Buffer 存放要写入的信息。读操作含两个 Msgs 信号 Msgs[0]和 Msgs[1]。需要注意的是，Buffer 在 Msgs[0]和 Msgs[1]中的功能是不同的：Msgs[0]存放读取的目标地址，Msgs[1]存放读取的内容。简而言之，I^2C 读操作分两步进行，首先需要将目标寄存器的地址写入，然后再从目标寄存器中读取内容并存放与 Buffer 中。I^2C 写操作是类似的，不过区别在于只含一个 Msgs 信号，即在写目标寄存器地址的同时对寄存器进行写操作。

3.4.5 OV3640 控制模块的实现

在完成 I^2C 读写函数的封装后，实现 OV3640 控制模块的工作就变得很容易了。这里给出 OV3640 控制模块的关键代码，以数码变焦为例，为实现数码变焦，需要同时修改寄存器 0x335F、0x3360 和 0x3361 的值。这里给出数码变焦的实现伪码：

```
I2cHandle = open("/dev/i2c/0", O_RDWR);  //开启 I2C 设备，获得句柄
AddrZoomHigh = 0x33;  //寄存器地址由 AddrZoomHigh 和 AddrZoomLowx 共同组成
AddrZoomLow1 = 0x5F, AddrZoomLow2 = 0x60, AddrZoomLow3 = 0x61;
//读取寄存器的值，存于 Reg
I2cRead(I2cHandle, AddrZoomHigh, AddrZoomLow1, &Reg1);
I2cRead(I2cHandle, AddrZoomHigh, AddrZoomLow2, &Reg2);
I2cRead(I2cHandle, AddrZoomHigh, AddrZoomLow3, &Reg3);
Width = ((Reg1&0x0F)<<8)+Reg2;  //根据 Reg 计算当前视频尺寸
Height = ((Reg1&0x70)<<4)+Reg3;
I2cWrite(I2cHandle, AddrZoomHight, AddrZoomLow1, DataSheet[Level][0]);
//DataSheet 为预设值，需要向寄存器中写入
//Level 为数码变焦的档数，OV3640 共支持五档
I2cWrite(I2cHandle, AddrZoomHight, AddrZoomLow2, DataSheet[Level][1]);
I2cWrite(I2cHandle, AddrZoomHight, AddrZoomLow3, DataSheet[Level][2]);
```

3.5 H.264 参数的优化设置

3.5.1 视频编码技术的发展

自上世纪八十年代数字电视信号诞生以来，图像压缩编码技术已有数十年的发展历史，该技术一方面在理论研究领域取得重要突破，另一方面在应用实践领域中也同样取得了令人瞩目的成果。图像编码技术在近些年里取得的发展令人瞩目，学术界和工业界对图像编码标准的制定标识着该技术日渐成熟和完善。譬如由国际标准化组织（ISO）和国际电工委员会（IEC）制定的关于静止图像的编码标准 JPEG/JPEG2000，关于活动图像的编码标准 MPEG-1、MPEG-2、MPEG-4 等，以及国际电信联盟（ITU）制定的视频编码标准 H.26X 系列。上述标准包含了当前最先进的多种编码算法，体现了图像编码技术的研究现状和成果。视频编码标准发展的出发点和根本目标是在确保有限的码率的前提下尽量提高图像质量[13]。伴随着日常生活对视频图像通信需求的提高，一个技术问题就突显出来，即如何使视频图像数据透明地在信道上传输而忽略信道的异构性。为解决上述难

题，由 ITU-T 和 ISO 共同提出的 H.264 编码标准应运而生。

3.5.2 H.264 视频编码标准

H.264 视频编码标准与传统视频编码标准均采用了基于块的混合编码方式。但 H.264 在此基础上新增了许多技术以提高其编码性能，主要包括：（1）采用先前已编码图像作为参考，在某些情况下允许多达 16 个参考帧，此特性使得 H.264 比传统视频编码标准更加灵活。相比之下，传统视频编码标准通常只允许一个或两个参考帧。此特性可以在一定程度上改善绝大多数场景下的比特率和图像质量。不过在某些特定场景下，如物体重复的动作或场景的反复切换或背景遮挡区域，H.264 能够在保持图像清晰的同时显著降低比特率；（2）采用可变大小块的运动补偿技术，块大小范围在 4×4 到 16×16 之间，从而使得对运动区域的分割更加精细。此外还支持多种尺寸的亮度预测块，并且可在同一宏块中混合使用不同尺寸的亮度预测块。色度预测块的尺寸由实际应用中的色度抽样所决定，一般较小；（3）在 B 宏块具有 16 个 4×4 分区的前提下，允许对该宏块采用多个（最高可达 32 个）运动矢量。每个 8×8 分区或更大的分区的运动矢量能够标识出参考图像之间的差异；（4）在 B 帧中可以使用任何形式的宏块（包括 I 宏块），使得使用 B 帧时的编码效率更高；（5）运动补偿精度可达双线性四分之一精度，从而提供更高精度的运动块预测；（6）加权预测。允许编码器在进行运动补偿时指定权重增加和偏移，同时在某些特殊场景下，如淡入、淡出、淡出后再淡入等情况下提供可观的增益。

相比于传统视频编码标准，H.264 能在一定程度上克服由于信道状况不佳（如信道丢包率高）导致的马赛克等现象。传统视频编码标准的分层结构如图 3-8 所示，这种结构与 TCP/IP 协议十分类似，每层包含头部信息和净荷部分，从而使得每层的头部信息与它的净荷部分之间形成了十分强的依赖关系。头部信息是该层的核心部分，若传输过程中头部信息丢失则收到的其余数据净荷部分也难以再恢复出正确的原始数据。尤其在序列层及图像层，由于 MTU 的限制，不可能将整个层的数据全部放在单个分组中，这时如果头部所在的分组丢失，该层其他分组即使能被正确接收也无法解码，造成带宽浪费。

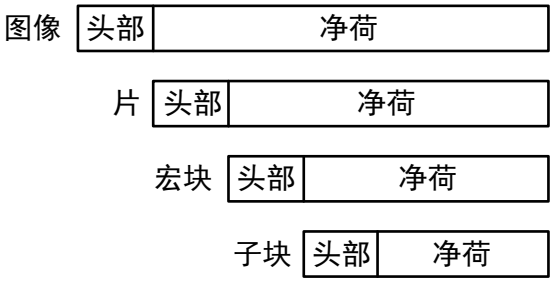

图 3-8　传统视频编码的分层结构

相比于传统视频编码的分层结构，H.264 的创新之处在于并未采用类 TCP/IP 的分层结构，换言之 H.264 取消了序列层和图像层，转而提出了序列参数集和图像参数集的概念用以存储关键编码信息，而将剩余相对不重要的信息存储于片层之内，如图 3-9 所示。序列参数集和图像参数集具有独立性，不依赖于片层。参数集与图像或序列之间是多对多的映射关系，即同一个参数集可以被多个序列中的图像参数集引用。同理，同一个图像参数集也可以被多个图像引用[14]。正是由于这一特性，在实现 H.264 时可建立针对上述两个参数集采取相应的保护机制。

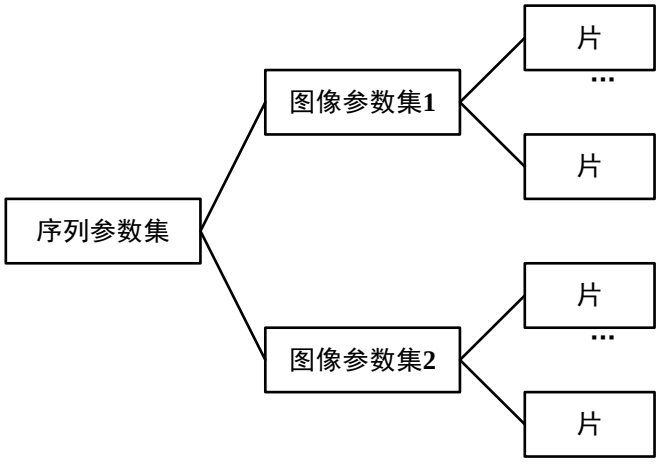

图 3-9　H.264 的分层结构

3.5.3　H.264 在无线环境中的应用

本文之所以选用 H.264 视频编码技术，一方面是由于 S3C6410 嵌入式平台内置了支持包括 H.264 在内多种硬件编解码标准的多媒体编解码器 MFC，能够在采用视频编码的同时，确保编码效率以满足多媒体传输的实时性要求，另一方面 H.264 编码技术自身的特性很适合应用于无线环境中，换言之，经过 H.264 编码后的数据能够透明地在多种无线网络协议下进行传输，具有较好的兼容性。除此之外，视频编码在会话应用中提高容错的支持也是很重要的，H.264 编码能很好地满足上述需求。H.264 在无线环境中的主要应用之一是移动多媒体电话服务，第三代移动通信合作组织 3GPP 已经在第六次发布中批准 H.264/AVC 作为其移动多媒体电话服务的可选技术。

3.5.4　GOP 和 QP 的优化设置

在实际应用中发现，虽然针对序列参数集和图像参数集等重要信息采取了保护机制，但使用 H.264 编码进行无线传输所获得的视频图像仍有马赛克现象，且图像失真度较高。分析后认为，产生上述问题的原因在于无线信道误码率较高导

致 I 帧丢失，进而造成同一 GOP 组的后续 P 帧无法解码从而产生马赛克现象。另一方面，默认的量化参数过低，这样虽能减轻带宽压力，但缺点是图像失真度较高。针对以上问题，本文提出如下解决方案：一是调整 GOP 参数，提高 I 帧比例，确保 I 帧丢失后能够及时刷新；二是调整 QP 参数，在保证一定带宽能力的前提下尽可能地获得精细的量化。

3.5.4.1 GOP

GOP 即图像组，表示一组连续的图像帧，图像组的结构如图 3-10 所示。

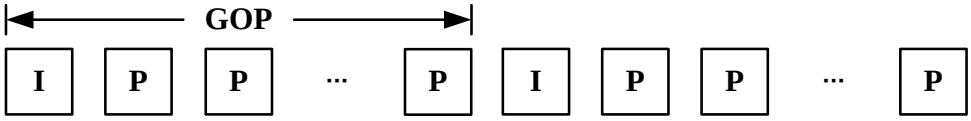

图 3-10 图像组结构

H.264 图像帧包括 I 帧和 P 帧两种类型，其中 I 帧是内部编码帧，包含一副图像的全部信息，携带信息量较多，而 P 帧是前向预测帧，只记录相对于同图像组的 I 帧的差值，携带信息量较少。因此虽然 P 帧相对 I 帧较小，但仅凭 P 帧无法完成对视频图像的解码，因为图像的关键信息存放于尺寸较大的 I 帧之中，GOP 参数即表示 I 帧与 P 帧之间的比例。例如，GOP=30 表示每 30 帧中的头一帧为 I 帧，随后 29 帧均是以该 I 帧为参考的 P 帧。此 30 帧结束后就将原有的 I 帧丢弃，并开始解码以下一 I 帧为首的新的图像组。由此可见，GOP 的值越小，I 帧出现的频率就越高，但对应的数据量也要增大，因为携带完整图像信息的 I 帧的数据量远大于只含差值信息的 P 帧。由于在 H.264 默认参数 GOP=30 情况下，视频的马赛克现象较为明显，故应考虑适当降低 GOP 参数的值，提高 I 帧比例，确保能够在丢包后及时刷新图像。

3.5.4.2 QP

QP 参数决定了 H.264 的量化精度。在数字信号处理领域，量化是指通过某种方式将一个较大的输入值集合映射为另一个较小的输出值集合。量化的方法一般是将某些单元值进行舍入，执行量化算法的设备或函数称之为量化器。由于量化而造成的误差一般称之为量化误差或舍入误差。量化有两个最主要的应用，一是对其他计算中获得或使用到的某些数值进行简单的近似，一般称之为舍入量化。二是在比特率受到信道或存储介质限制的情况下，需要在压缩数据的同时对其失真程度进行控制，此时就需要用到量化，此时一般称量化为率失真优化。任何有损压缩算法的核心基本上都涉及到了量化。本节所讲的 QP 属于量化的后一种应用。H.264 可以在几乎不影响用户视觉体验效果的前提下降低图像编码长度，压

缩原始图像中携带的数据冗余。H.264 采用标量量化技术，将每个图像样点编码映射为较小的数值，如式（3-1）所示：

$$Y = round(X / \mathrm{QP}) \qquad \text{式（3-1）}$$

其中 X 为原始图像的输入样本点编码；QP 为量化指标，QP 的取值范围在 0 到 51 之间，其中 0 代表量化精细度最高，51 代表量化精细度最低；Y 为 X 的输出量化值。由于在 H.264 默认参数 QP=26 情况下，视频图像失真度过高，故应考虑适当降低 QP 参数的值，在保证带宽能力的前提下提高量化精度。

3.6 视频采集的实现

在优化了 H.264 视频编码的若干参数后，就可以进行视频数据的采集，并将采集到的原始 YUV 视频数据送至 H.264 编码器进行视频编码压缩，编码后的数据被送至传输模块进行进一步的处理。这里给出视频采集的实现伪码：

```
CameraHandle = CameraCaptureInit();   //获得 OV3640 视频传感器句柄
//启动 H.264 编码器
EncoderHandle = MfcEncoderInit(LcdWidth, LcdHeight, 30, 1000, 30);
Value[0] = H264EncParamGopNum;
Value[1] = 5;
SsbSipH264EncodeSetConfig(EncoderHandle, ConfType, Value);   //将 GOP 设为 5
Value[0] = H264EncParamIntraQp;
Value[1] = 25;
SsbSipH264EncodeSetConfig(EncoderHandle, ConfType, Value);   //将 QP 设为 25
while(True){
    Fread(YuvBuffer, 1, YuvBufferSize, CameraHandle);   //采集原始 YUV 视频数据
    //将原始 YUV 数据送至 H.264 编码器进行编码
    EncodedBuffer =
    MfcEncoderExe(EncoderHandle, YuvBuffer, YuvBufferSize, 1,&EncodedBufferSize);
    ……
}
```

3.7 音频采集的实现

3.7.1 ALSA 简介

为使多媒体传感节点能正常采集音频数据，本文需要为 S3C6410 配置合适的音频驱动。目前 Linux 中常用的音频驱动有 OSS 和 ALSA 两种。OSS 即 Open Sound System，是最早出现在 Linux 上的音频编程接口。OSS 的优点是发展成熟，

但作为一个商业产品，其非开源特性并不为 Linux 内核开发者们所支持。从 Linux 2.5 版本开始 ALSA 被引入。ALSA 表示高级 Linux 声音体系结构（Advanced Linux Sound Architecture），是为声卡提供驱动的 Linux 内核组件，用以替代传统的 OSS。ALSA 由一系列内核驱动、应用程序编程接口（API）以及一组实用的 Linux 音频程序套件组成，相比于 OSS 下直接使用 ioctl 编程，ALSA 所提供 API 屏蔽了底层实现细节，能够极大地提高开发效率，其体系结构如图 3-11 所示。

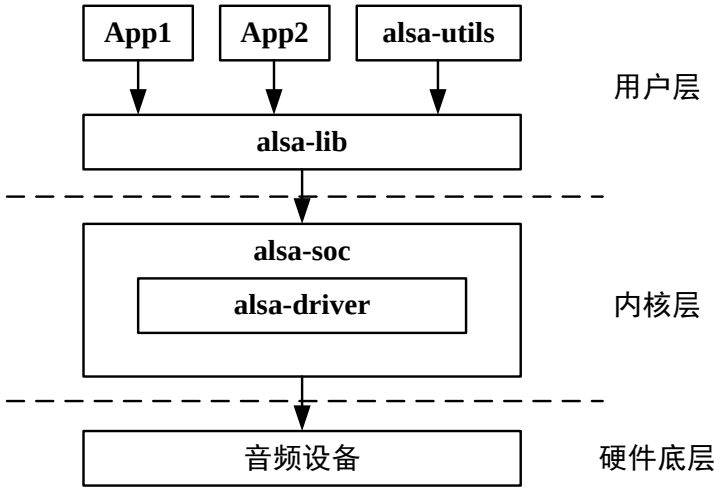

图 3-11　ALSA 体系结构

由图 3-11 可以看到，用户层的 alsa-lib 对应用程序提供统一的 API 以便隐藏底层的实现细节，本文将在 3.7.3 节利用 alsa-lib 实现音频采集模块，此处暂不赘述。ALSA 所提供 API 包括：（1）控制接口，该接口用于管理声卡的寄存器和查询可用设备；（2）PCM 接口，该接口用于用于管理数字音频的采集与回放，是最常用的接口之一；（3）原始 MIDI 接口，该接口支持 MIDI（Musical Instrument Digital Interface，乐器数字接口）并提供了对声卡中 MIDI 总线的支持。利用该接口，开发人员只需负责管理协议和时序，而将底层的 MIDI 事件交由接口进行管理；（4）计时器接口，该接口用于对声卡中的硬件进行计时以便同步多个音频事件；（5）音序器接口，与原始 MIDI 接口相比，该接口是一个更高层次的 MIDI 编程与声音合成接口，通过该接口能处理大量的 MIDI 协议和时序。（6）混音器接口，该接口用于控制设备的音效卡，包括音量的控制等，属于最顶层的控制接口。此外 ALSA 还提供了基于 alsa-lib 开发的一系列 Linux 音频应用程序套件，称之为 alsa-utils。alsa-utils 主要包括：（1）alsactl，用于对 ALSA 声卡驱动进行一些高级设置，譬如对系统中多个声卡的配置。有时无法在音量控制面板进行调整时也可以通过 alsactl 来实现控制；（2）aconnect，作为 ALSA 音序器的连接管理器，一般用于连接或断开 ALSA 音序器上的端口，端口可随意定义；（3）amidi，用于对 ALSA 的原始 MIDI 端口进行读写操作；（4）amixer，是 ALSA 声卡驱动

设置命令行工具，由于 S3C6410 不支持 ALSA 图形界面控制面板 alsamixer，因此主要通过 amixer 来对 ALSA 驱动参数进行设置。在本文 3.7.2 节移植 ALSA 驱动模块中需要用到 amixer；（5）arecord 和 aplay，是命令行的 ALSA 声卡驱动的录音和播放工具，本文在 3.7.2 节会使用它们测试 ALSA 驱动模块是否移植成功。在图 3-11 的内核层中的 alsa-soc 是对 alsa-driver 的进一步封装，它针对嵌入式设备提供了一系列增强功能。综上，内核层的 alsa-driver 是 ALSA 最基本最底层的部分，alsa-utils 和 alsa-lib 函数库均是以此为基础，因此下一步本文将对 ALSA 驱动模块进行移植。

3.7.2 ALSA 驱动模块的移植

S3C6410 并不原生支持 ALSA 驱动，因此需要将 ALSA 源码交叉编译后部署至 S3C6410，步骤如下：

（1）交叉编译 alsa-lib

```
./configure --hLinuxt=arm-none-linux-gnueabi --prefix=/usr/share/arm-alsa
```

（2）将 PC 端挂载至 S3C6410 的/mnt 目录下后将已编译好的 alsa-lib 部署至 S3C6410

```
cp –r /mnt /arm-alsa/usr/share
cp –r /usr/share/arm-alsa/lib/* /lib
```

（3）编辑启动脚本/etc/init.d/rcS，添加如下几行

```
mkdir /dev/snd
ln /dev/controlC0 /dev/snd/controlC0
ln /dev/pcmC0D0c /dev/snd/pcmC0D0c
ln /dev/pcmC0D0p /dev/snd/pcmC0D0p
ln /dev/timer /dev/snd/timer
amixer cset numid=86 2
amixer cest numid=87 2
```

（4）编辑/etc/profile，添加如下几行

```
export ALSA_CONFIG_PATH=/usr/share/arm-alsa/share/alsa/alsa.conf
```

（5）移植完毕，为测试 ALSA 驱动是否移植成功，用 ALSA 提供的应用程序 arecord 录制一段音频数据，时长为 60s，采样位数为 16bit，采样率为 8000，单声道，封装格式为 wav。然后用 aplay 播放查看录制是否成功。使用如下两行命令

```
arecord –fS16_LE –r 8000 –t wav –c 1 –d 60 test.wav
aplay test.wav
```

3.7.3 音频采集的实现

ALSA 驱动模块移植完毕后，就可以利用其提供的 API 实现音频采集功能，这里给出音频采集的实现伪码：

```
//打开音频设备，设备句柄存放于 AudioHandle
SndPcmOpen(&AudioHandle, “default”, SND_PCM_STREAM_CAPTURE, 0);
//为 AudioParams 分配存储空间，该变量用于存放 ALSA 各项参数，包括采样率、
//采样位数、声道数等
SndPcmHwParamsMalloc(&AudioParams);
//将音频采集方式设为交错模式
SndPcmHwParamsSetAccess(AudioHandle, AudioParams,
SND_PCM_ACCESS_RW_INTERLEAVED);
//将采样位数设为 16Bit
SndPcmHwParamsSetFormat(AudioHandle,
AudioParams, SND_PCM_FORMAT_S16_LE);
//将采样频率设为 8000Hz
SndPcmHwParamsSetRateNear(AudioHandle, AudioParams, 8000, 0);
//将声道数设为 1，即单声道
SndPcmHwParamsSetChannels(AudioHandle, AudioParams, 1);
//初始化设备，准备采集
SndPcmPrepare(AudioHandle);
while(True){   //采集
    SndPcmReadi(AudioHandle, PcmBuffer, PeriodSize);
    ……
}
```

第四章　传输模块的设计与实现

4.1　传输模块的总体设计

本文所设计实现的传输模块主要实现以下功能：

（1）在采集节点进行数据的信道编码和发送，具体为：在接收到采集模块所提供的视频数据和音频数据后，对数据分组并对视频数据进行 LT 信道编码，然后将数据封装，主要是添加包括分组长度、分组序号、LT 编码信息等必要的首部信息，最后通过 socket 进行传输；

（2）在接收节点进行数据的 LT 解码和最优分组长度实时反馈机制。具体包括：1）对接收到的数据进行 LT 解码；2）利用基于 Gilbert 模型的信道丢包率统计模块获取当前信道状况，依据该信息计算当前信道状况下发送端应采用的最优分组长度（或称为分组长度），将该信息反馈给发送端，从而减少 LT 编码带来的额外冗余数据，降低带宽压力。传输模块的总体设计如图 4-1 所示。

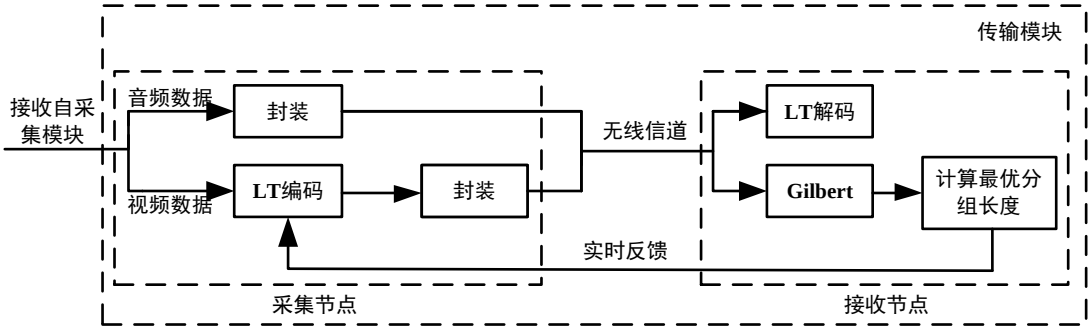

图 4-1　传输模块总体设计

如图 4-1 所示，实现传输模块需要完成的工作主要有以下几部分：

（1）实现 LT 前向信道编解码。此项工作的主要目的在于提高系统的抗干扰性，在信道状况不佳的情况下，可能会出现数据分组丢失的情况，若不采取任何保护措施则该分组对应的原始数据将无法恢复出来，并且隶属于该原始数据的其余分组也将变得无用，成为浪费带宽的“垃圾分组”。因此实现 LT 编解码对提高多媒体数据传输质量具有重要意义。另一方面，LT 编解码虽能有效提高系统抗干扰性，但其需要消耗一定的处理器资源，因此为了在系统抗干扰性与多媒体通信实时性之间取得平衡，本文仅对视频数据采用了 LT 编码，而对音频数据采取直接传输的策略。这也是考虑到音频数据对网络性能要求相对较低，在较差信道状况下仍能满足用户体验。此部分工作焦点在于如何实现有效、稳定、可靠的 LT 编解码模块。

（2）实现 Gilbert 网络统计模块。本文的目标之一是为传输模块实现实时反

馈最优分组长度机制，而反馈的依据是实时的信道状况，具体为网络的实时误码率或丢包率。相较于传统的伯努利模型，Gilbert 模型统计更准确，更能反映无线信道的记忆特性。也就是说，Gilbert 模块是为后续的最优分组长度实时反馈机制服务的，是实现该机制的前提。此部分工作焦点在于如何实现有效、准确的 Gilbert 模块。

（3）设计并实现最优分组长度实时反馈机制。LT 前向纠错编码虽然能提高系统的抗误码能力，但另一方面也提高了数据的冗余量，为本就不宽裕的无线信道的带宽增加了压力。为了克服 LT 编码为系统带来的这一负面影响，本文考虑在保证抗误码能力的基础上尽可能降低编码冗余。此部分的工作焦点在于如何设计有效的数学模型，能够准确地反映最优分组长度与信道丢包率之间的关系，并对此模型进行代码实现，使其能够在通过 Gilbert 模型获得当前信道丢包率后，快速、准确地计算出此信道状况下应采用的最优分组长度，并将该信息反馈至采集节点的 LT 编码模块，达到降低 LT 编码数据冗余的效果。

（4）数据的封装与传输。采集节点在完成 LT 编码后，需要对数据进行必要的封装，包括分组信息、LT 编码信息等，在封装完成后通过 socket 将数据发送出去。因此本部分工作焦点在于如何设计有效的首部信息，在确保接收节点利用此信息能正确地恢复出原始数据的同时，尽可能降低首部信息的空间开销。

4.2 LT 前向信道编码

4.2.1 数字通信系统

数字通信系统是以数字信号作为传输对象的一种通信系统[15]，本文所研究的传输模块就属于数字通信系统的一种。一个完整的数字通信系统模型如图 4-2 所示。

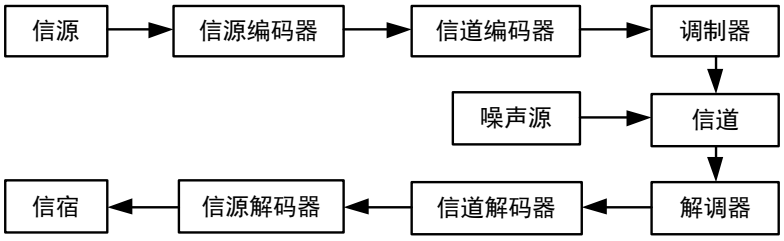

图 4-2　数字通信系统模型

信源是整个系统信息的来源，即信息的发布者，它可以提供多种类型的信息，如声音、图像、文字等。信源编码器接收到信源提供的各种信息后，将这些信息统一转化为数字信号以便后续进行传输，此过程称为模/数转换。在必要时信源编码器还会使用某些编码算法对数字信号进行压缩，以降低数据冗余，本文在 3.5 节中提到的 H.264 视频编码技术就属于信源编码的一种。在经过信源编码器

的模/数转换和压缩编码后，为提高系统的抗干扰性，还可将数据继续送至信道编码器。该模块的原理是在需要传输的数据中增加一些冗余信息，使得在接收端能够根据这些信息进行检错甚至纠错。调制器对数据做一些必要处理，使之适合在信道上传输。随后数据被送至信道，接收端在收到数据并进行解调后，将数据送至信道解码器。由于实际信道中总会存在一些噪声，因此接收端收到的数据会或多或少地产生部分失真。信道解码器的作用就是利用冗余信息对失真信号进行检错甚至纠错。可以说信道解码器是信道编码器的逆过程，信源解码器亦然。最后经过信源解码器解码获得的原始数据被送至信宿，即信息的接受者。整个数字通信系统的工作流程至此结束。

4.2.2 前向信道编码和 LT 编码

前向信道编码属于信道编码的一种，由上一节可知信道编码需要为待传输的数据增加一定的冗余以便在接收端进行检错甚至纠错。其理论依据是 1948 由美国数学家香农提出的有噪信道编码定理，即香农定理。香农定理的基本思想是对任意有噪声信道，设其信道容量为 C，其定义如式（4-1）所示：

$$C = B \cdot \log_2(1 + S/N) \qquad \text{式（4-1）}$$

其中 B 表示信道带宽，S/N 表示信道的信噪比。设在该信道上的数据传输速率为 R。当 $R < C$ 时，总能找到一种信道编码技术使得接收端收到数据的错误无限小，即能实现无差错传输。而当 $R > C$ 时，不能保证无差错传输。因此信道编码的最终目标是逼近香农限制。

差错控制编码已经经历了 50 多年的发展，从早期的 BCH 码和 RS 码，到后来的 LDPC 码、Turbo 码以及卷积码，再到近些年出现的数字喷泉码，差错控制编码的发展脚步从未停下，并且应用范围越来越广[16]。在本文中，由于多媒体数据尤其是视频数据的数据量较大，同时实时性要求较高，因此需要有效的传输机制进行传输。但另一方面，WMSNs 所采用的无线信道自身的带宽能力和抗干扰性都很有限，若不经任何处理而直接对多媒体数据进行传输很可能造成传输质量不佳的结果。对此，本文考虑引入数字喷泉码中较为成熟的代表 LT 前向信道编码，将其纳入系统的传输模块中，以期提高系统的抗干扰性。

LT 前向信道编码的主要思想是随机编码，其码率并不像传统编码那样固定不变，而是在传输过程中动态可变的。采用 LT 前向纠错编码后，接收端在收到任意一组固定数量（一般是稍多于原始数据分组个数，多出的部分称为编码开销）的编码分组之后，就能将所有的原始数据分组正确地恢复出来，使系统具备一定的纠错能力。与传统的差错控制编码相比，LT 前向信道编码的优势在于：（1）编解码复杂度较低，尤其是在异构信道情况下；（2）采用 LT 前向信道编码后，

在整个的传输过程中无需再采用任何的重传机制，因此能有效降低对带宽资源的占用。对于 LT 前向信道编码来说，度分布的设计是影响其性能的一个关键。目前学术界提出了众多度分布设计思路，包括理想孤立子度分布、稳健孤立子度分布[8]、次优度分布[17]等。本文经过综合考虑比较，选取了较为成熟和较为容易实现的稳健孤立子度分布作为 LT 前向信道编码的度分布函数，对此将在本文的 4.2.3.2 节进行详细的阐述。

4.2.3 LT 编码原理及实现

4.2.3.1 LT 编码原理

LT 编码过程如下：

（1）将原始数据按长度 *L* 切分为等长的 *K* 个分组，称为原始分组。若最后一片长度不足则将其长度填充至 *L*。

（2）依据给定的度分布函数随机选择一个度 *D*。

（3）随机从原始分组中选择 *D* 个原始分组，对这 *D* 个原始分组进行异或，异或的结果就是一次 LT 编码的结果，称之为编码分组。

（4）重复步骤（2）和（3）就可以源源不断地产生编码分组。

由此可见，LT 编码的核心是度分布函数，度分布函数的选择直接决定了 LT 编码的性能，较优的度分布函数能够让 LT 解码器在获得尽量少的编码分组后就能解码出原始数据。

4.2.3.2 度分布函数

根据 LT 前向信道编码的特性可知，解码端在收到任意一组固定数量（一般是稍多于原始数据分组个数，多出的部分称为编码开销）的编码分组之后，就能将所有的原始数据分组正确地恢复出来，假设此数量为 *K*’。由 *K* 个原始分组和 *K*’个编码分组，可以得出一个二分关系图，如图 4-3 所示。

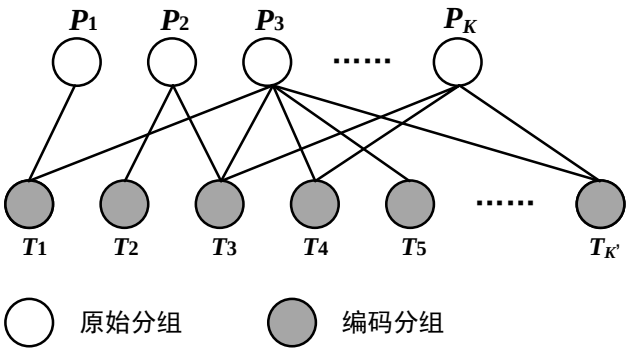

图 4-3　LT 编码二分图

度分布函数本质上是一个概率分布函数 $\mu(i)$，其函数值表示度为 i 的概率。目前学术界提出许多度分布函数的构造方法，包括理想孤立子度分布函数、稳健孤立子度分布函数[8]、次优度分布[17]、基于 NSD 度分布函数的 LT 码构造[18]等。本文采用稳健孤立子度分布函数，此度分布函数算法简单，适用于计算能力较差的嵌入式平台，且构造效果较好，能够满足本文的需求。稳健孤立子度分布函数的构造方法如下：

（1）给定常数 $R=2+8\sqrt[4]{K}$，R+1 对应预处理集（Ripple）的初始长度，预处理集会在后续 LT 解码处进行介绍。

（2）给定变量 δ 表示恢复出原始数据的失败概率，则有如下关系。

$$K_1 = K + O(\ln^2(K/\delta)\cdot\sqrt{K}) \qquad \text{式（4-2）}$$

（3）在给定 R 和 δ 之后，可以分别定义 $\rho(i)$和 $\tau(i)$如下，其中 $\rho(i)$为理想孤立子度分布函数。

$$\rho(i)=\begin{cases}\dfrac{1}{K}, & i=1\\ \dfrac{1}{i\cdot(i-1)}, & i=2,\cdots,K\end{cases} \qquad \text{式（4-3）}$$

$$\tau(i)=\begin{cases}\dfrac{R}{i\cdot K}, & i=1,\cdots,\dfrac{K}{R}-1\\ \dfrac{R\cdot\ln(R/\delta)}{K}, & i=\dfrac{K}{R}\\ 0, & i=\dfrac{K}{R}+1,\cdots,K\end{cases} \qquad \text{式（4-4）}$$

（4）将 $\rho(i)$和 $\tau(i)$作归一化处理，得到所求的稳健孤立子度分布 $\mu(i)$，其中 β为衡量 LT 编码数据冗余的参数，β值越大，表示冗余量越大。此项参数在后续设计最优分组长度模型时会用到。

$$\beta=\sum_i(\rho(i)+\tau(i)) \qquad \text{式（4-5）}$$

$$\mu(i)=\frac{\rho(i)+\tau(i)}{\beta}, \quad i=1,2,\cdots K \qquad \text{式（4-6）}$$

4.2.3.3 有限随机 LT 码构造

从本文 4.2.3.1 节的 LT 编码过程中可以看到，每个编码分组对应的 D 个原始分组（或称为邻居）均是随机生成的，且这种随机生成过程是相互独立的。这种编码方式会存在两个问题：首先，随机数生成算法复杂度较高，若对每个编码分组的每个邻居都调用一次随机数生成算法，会造成较大的时间开销，对于计算能力相对较弱的嵌入式平台来说是值得商榷的；其次，若随机生成过程是彼此独立

的，则在传输过程中需要使编码分组携带所有邻居信息，以便能在解码端获取足够的信息进行解码。但是每个编码分组的度不同，对此必须采用变长的数据包首部，这会造成数据结构设计的复杂化，且若编码分组的度较大，会产生很大的首部开销，降低传输效率。

本文采用有限随机 LT 码构造来解决上述问题。其方法是对每个编码分组只生成两个随机数 a 和 b，而此编码分组的所有邻居编号由 a 和 b 通过线性同余法生成[19]。通过这种方法可以降低随机数生成算法的调用频率，同时在传输时只需要传输 a 和 b 的值，而在解码端使用同样的线性同余法即可获得其所有邻居编号，使得数据结构设计变得简单，且无论编码分组的度为多少均不会产生很大的首部开销。

线性同余法的递推公式如下，其中 x_i 为编码分组的第 i 个邻居的编号，其值范围为[0, K)，x_0 为初始值且 $0 \leqslant x_0 < K$。

$$x_i = (a \cdot x_{i-1} + b) \bmod (K) \qquad \text{式（4-7）}$$

4.2.3.4 LT 编码的实现

下面给出 LT 编码的实现伪码：

```
K = ceil((double)F/T);      //获取每帧的分片数
R = 2+8*pow(K, 0.25);       //根据分片数，计算常量R
K1 = DgrGen(K);   //生成度分布函数及 K'的值
for(i = 1; i <= K1; i++){
    A = Random(i);
    B = Random(i);
    Dgr = GetDgr();   //通过度分布函数随机获得此编码分组的度
    for(j = 1; j <= Dgr; j++){
        if(j == 1){
            Index = Random(i)%K;
            EncSymbol[i] = SrcSymbol[Index];  //得到第一个分片
        }
        else{
            Index = (A*Index + B)%K;   //线性同余法随机生成邻居编号
            EncSymbol[i] ^= SrcSymbol[Index];   //编码
        }
    }
}
```

4.2.4 LT 解码原理及实现

4.2.4.1 LT 解码原理

LT 解码原理如下：

（1）初始状态下 K 个原始分组均未被恢复。

（2）找出所有度为 1 的编码分组，恢复出其对应的原始分组。实际上度为 1 的编码分组和其对应的原始分组是相同的。将恢复出的原始分组加入预处理集。

（3）从预处理集中选取一个已恢复出的原始分组，将其与二分图中与其相关的编码分组进行异或，并将所有这些编码分组的度减 1，同时从预处理集中删除这个原始分组。

（4）重复步骤（2）和（3），直至所有 K 个原始分组均已被恢复或未完全恢复但预处理集大小已变为 0。前者表示解码成功，后者表示解码失败。

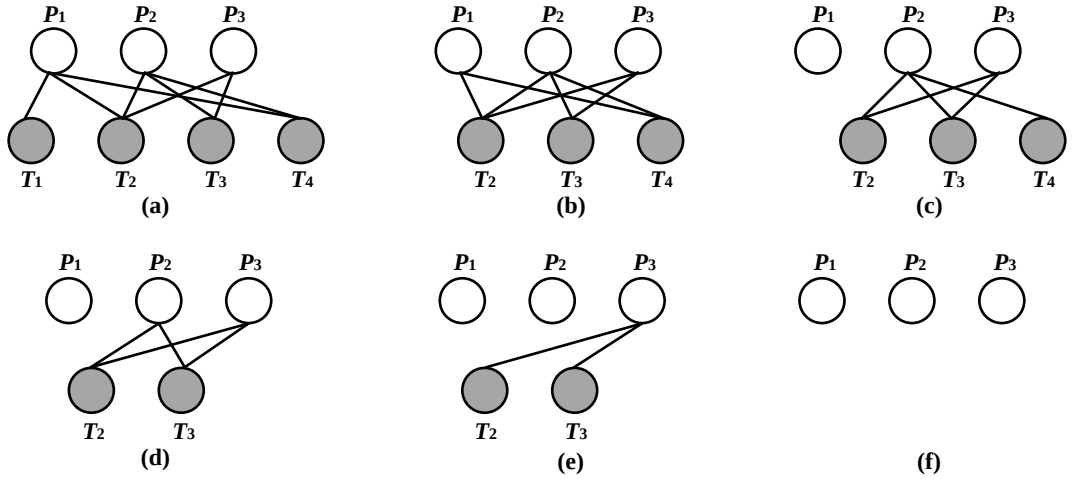

图 4-4　LT 解码过程

以图 4-4 的例子详细说明 LT 解码过程：

（1）初始状态下有三个原始分组$\{P_1, P_2, P_3\}$和四个编码分组$\{T_1, T_2, T_3, T_4\}$，三个原始分组均为待解码状态，如图 4-4(a)所示。

（2）找到度为 1 的编码分组即 T_1，恢复出其对应的原始分组 P_1。由异或的特性易知有 P_1=T_1，将 P_1 加入预处理集并删除 T_1 及 T_1 与 P_1 相连的边，如图 4-4(b)所示。

（3）从预处理集中取出 P_1，将其同剩余与其相关的编码分组 T_2 和 T_4 进行异或并删除相连的边，同时 T_2 和 T_4 的度减 1。此时编码分组 T_4 的度降为 1，如图 4-4(c)所示。

（4）由 T_4 恢复出 P_2 并将 P_2 加入预处理集，同时删除 T_4 及 T_4 与 P_2 相连的边，如图 4-4(d)所示。

（5）从预处理集中取出 P_2，将其同剩余与其相关的编码分组 T_2 和 T_3 进行异或，并删除相连的边，同时 T_2 和 T_3 的度减 1。此时 T_2 和 T_3 的度均降为 1，因此可恢复出原始分组 P_3，如图 4-4(e)所示。

（6）至此，原始分组$\{P_1, P_2, P_3\}$均已被恢复出来，解码过程完毕，如图 4-4(f)所示。

LT 解码的逻辑流程如图 4-5 所示：

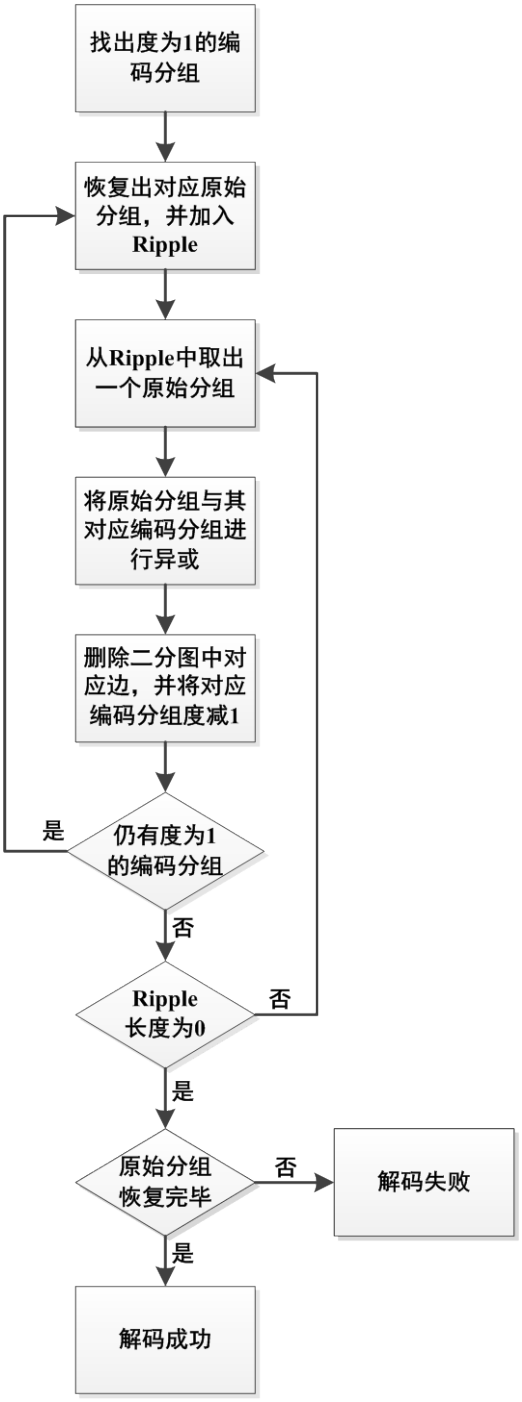

图 4-5　LT 解码流程

4.2.4.2 LT 解码的实现

下面给出 LT 解码的实现伪码：

```
EncSymbolTmp = GetDgr1();  //初始化，找出度为 1 的编码分组
SrcSymbolTmp = EncSymbolTmp;  //恢复出对应原始分组并加入预处理集
EnQueue(Ripple, SrcSymbolTmp); //预处理集为队列
while(!IsEmpty(Ripple)){  //预处理集为空则解码结束
    SrcSymbolTmp = DeQueue(Ripple);  //从预处理集中选择一个原始分组
    while((EncSymbolTmp = FindNextNeighbor(SrcSymbolTmp))){
        EncSymbolTmp ^= SrcSymbolTmp;  //异或解码
        Dgr[EncSymbolTmp]--;
    }
    while((EncSymbolTmp = GetDgr1())){  //将新的度为 1 的编码分组对应的
        SrcSymbolTmp = EncSymbolTmp;  //原始分组加入预处理集
        EnQueue(Ripple, SrcSymbolTmp);
    }
}
```

4.3 信道丢包率统计模块

本文采用基于 Gilbert 模型实现的信道状况统计模块进行对信道丢包率的统计，为后续实现最优分组长度实时反馈机制奠定基础。在最优分组长度实时反馈机制中，需要依据信道丢包率来计算当前信道状况下应采用的分组长度，以降低 LT 编码带来的数据冗余，因此需要信道丢包率统计模块具有较高的准确度，否则会间接影响最优分组长度实时反馈机制的有效性。

4.3.1 传统伯努利模型

记信道丢包率为 P_{lost}，其值越高则表示信道状况越差。在传统的伯努利模型中，仅仅是用已传输数据包数量和已接收到数据包数量来计算信道丢包率，即

$$P_{\text{lost}} = \frac{\text{已接收数据包数}}{\text{已发送数据包总数}} \qquad \text{式（4-8）}$$

伯努利模型的优点是简单、易于理解，这种以平均丢包率表示信道状况的思想在传统有线信道中的准确率尚可,但在本文所采用的无线信道中伯努利模型难以达到准确度要求。其原因在于无线信道属于有记忆型信道，具体来说无线信道中相邻的若干个丢包事件之间具有一定的相关性，而伯努利模型不能反映这种相关性，从而导致准确度下降。

4.3.2 Gilbert 模型原理

为反映无线信道中相邻丢包之间的相关性，本文引入了 Gilbert 模型。Gilbert 模型于 1960 年由 E.N.Gilbert 提出[20]，该模型采用了一个两状态的马尔科夫模型，常用于描述信道中的突发丢包状况[21][22]。在 Gilbert 模型中，定义一个随机变量 Z 表示事件“数据包是否丢失”，即

$$Z=\begin{cases}1, & \text{数据包丢失}\\ 0, & \text{数据包正确收到}\end{cases} \qquad \text{式（4-9）}$$

则 Gilbert 模型可用图 4-6 表示：

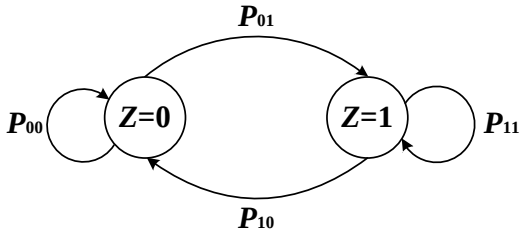

图 4-6 Gilbert 模型

如图 4-6 所示，用四个变量 P_{00}、P_{01}、P_{10}、P_{11} 分别表示网络在 Z=0 和 Z=1 两种状态间转换的四种情况的概率，其中 P_{01} 表示从 Z=0 状态转换到 Z=1 状态的概率，其余类似。由此可得出信道丢包率 P_{lost}

$$P_{\text{lost}}=\frac{\dfrac{P_{01}}{P_{01}+P_{00}}}{\dfrac{P_{01}}{P_{01}+P_{00}}+\dfrac{P_{10}}{P_{10}+P_{11}}} \qquad \text{式（4-10）}$$

4.3.3 信道丢包率统计模块的实现

根据式（4-10）可知，信道丢包率 P_{lost} 可由状态转换次数 N_{00}、N_{01}、N_{10}、N_{11} 计算得出。下面给出基于 Gilbert 模型的信道丢包率统计模块的实现伪码：

```
while(True){
    switch(PrePackageLost){
        case 0:   if(CurPackageLost == 0){ N00++; break; }
                  else{ N01++; break; }
        case 1:   if(CurPakageLost == 0){ N10++; break; }
                  else{ N11++; break; }
    }
    PLost = N01/(N01+N11)/(N01/(N01+N11)+N10/(N10+N11));
}
```

4.4 最优分组长度实时反馈机制

4.4.1 计算最优分组长度

根据 4.2.3.2 节中提到的稳健孤立子度分布的特性可知，衡量 LT 编码数据冗余量的参数 β 与 LT 编码输入分组数量 K 有关。具体来说，K 值越大则 β 值越小，即数据冗余越少；K 值越小则 β 值越大，即数据冗余越多。因此可以考虑尽可能地提高 K 值即分组数量。在本文中，经过 H.264 视频编码后的视频帧的长度 F 是在一定区间范围内的，无法进行改变，因此只能考虑降低分组长度 L 从而达到提高分组数量 K 的目的。但是另一方面，分组需要携带一定长度的首部信息以便于解码端正确恢复出原始数据，该首部信息的长度是固定的，因此降低分组长度的行为虽然能间接地降低 LT 编码的冗余，但同时也提高了单个分组内部首部信息所占比例，从而降低了净荷在分组中所占比例。综上所述，降低 LT 编码数据冗余和提高净荷比例之间存在一定矛盾。本文研究内容之一就是找出确定的分组长度，以平衡上述二者之间的关系，这样的分组长度称之为最优分组长度。

为计算最优分组长度，本文给出如下数学模型：

$$Y=\frac{\beta}{1-P_{\text{lost}}}\cdot\frac{H+L}{L}=\frac{\beta}{(1-E)^{H+L}}\cdot\frac{H+L}{L} \qquad \text{式（4-11）}$$

其中 Y 表示效用值，它衡量了 LT 数据冗余量与数据分组中净荷比例之间的平衡关系，它的值越低则表示平衡性越好。β 表示 LT 编码输出的编码分组个数 K'与 LT 编码输入的原始分组个数 K 的比值，该参数衡量了 LT 编码的数据冗余量，由式（4-5）可知在稳健孤立子度分布中 β 的值是由 K 唯一决定的，即 $\beta=\beta(K)$。P_{lost} 表示信道的丢包率，因此 $\frac{\beta}{1-P_{\text{lost}}}$ 表示发送端实际需要发送的数据冗余量，达到该值能确保 LT 解码端正确恢复出原始数据。H 表示分组内部的首部信息长度（分组首部的具体结构会在 4.5 节中给出），L 表示净荷长度，$\frac{H+L}{L}$ 表示净荷在整个分组中所占比例。E 表示信道误码率，这里用 E 替代 P_{lost} 是为了体现出分组长度 L 对信道丢包率的影响。根据式（4-11）可知，在给定信道丢包率 P_{lost} 的前提下，效用值 Y 由分组长度 L 唯一决定，即 $Y=Y(L)$。因此计算最优分组长度实际上就是在给定丢包率 P_{lost} 的前提下，求出确定的分组长度 L，使得效用值 Y 最小。

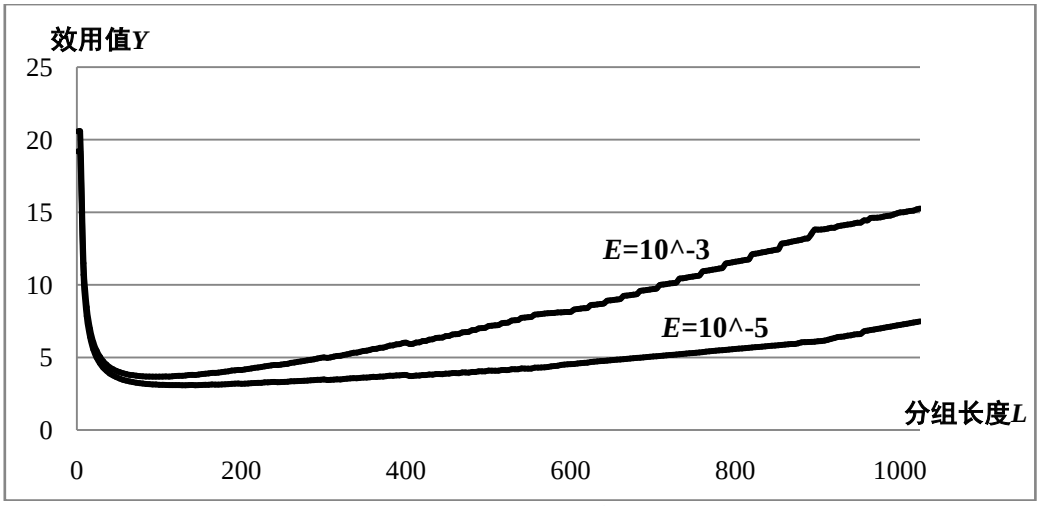

图 4-7　效用值 Y 与分组长度 L 的关系曲线

图 4-7 给出了误码率 E 分别为 10^{-3} 和 10^{-5} 情况下效用值 Y 与分组长度 L 之间的关系曲线，可以看出在不同误码率下均能取得某个确定的分组长度 L 使得效用值 Y 最小，此即本文所求的最优分组长度。

4.4.2　最优分组长度实时反馈机制的实现

在利用式（4-11）计算出给定误码率下的最优分组长度后，就可利用该数据实现最优分组长度的实时反馈机制。此机制的总体设计思路如图 4-1 所示。具体来说，接收节点会周期性地利用基于 Gilbert 模型的信道丢包率统计模块获取当前信道的丢包率 P_{lost}，并根据公式 $1-P_{\text{lost}}=(1-E)^{H+L}$ 计算出当前信道的误码率 E，从而得到该误码率下的最优分组长度 L_{best}，并将 L_{best} 的值反馈给发送端的 LT 编码模块处。LT 编码端获得该信息后，相应地修改输入 LT 编码器的原始分组的的长度，从而达到降低 LT 编码数据冗余量的目的。下面给出最优分组长度实时反馈机制中接收端实现的伪码（发送端只需在收到反馈信息后简单地修改原始分组长度即可，此处不再赘述）：

```
while(True){
    Wait(SatPeriod);   //周期性反馈
    PLost = GetPLost();   //获得信道丢包率 P_Lost
    E = 1-pow(1-PLost, 1/(double)(H+L));   //计算出信道误码率 E
    LBest = LBestCal(E);   //计算出最优分组长度 L_best
    SendTo(SendingEnd, Lbest);   //反馈至发送端
}
```

4.5 分组首部结构与分组传输

4.5.1 分组首部结构

本文最终是以分组为基本单位对数据进行传输的。所谓分组就是对采集到的视频帧和音频帧进行等长切分后得到的固定长度的数据（最后一个分组的长度可能小于该固定长度）。设计较好的分组首部结构能够在确保接收端能正确恢复出原始数据的同时尽可能地压缩分组首部开销。本文对视频数据和音频数据分别采用了如下分组首部结构：

```
typedef struct{    //视频数据的分组首部结构
    unsigned shortFrameType;   //视频帧？音频帧
    unsigned int FrameNo;      //帧编号
    unsigned long SliceNo;     //分片编号
    unsigned int FrameSize;    //帧大小
    unsigned int SliceSize;    //分片大小
    unsigned short K;          //原始分片个数
    unsigned short K1;         //LT编码后分片个数
    unsigned shortESI;         //LT编码后分片编号
    unsigned shortDgr;         //LT编码后分片度
    unsigned shortNumA;        //恢复原始分片的方法的系数
    unsigned shortNumB;        //同上
} VideoFrameHeader;

typedef struct{    //音频数据的分组首部结构
    unsigned short FrameType;
    unsigned int FrameNo;
    unsigned long SliceNo;
    unsigned int FrameSize;
    unsigned int SliceSize;
} AudioFrameHeader;
```

其中，成员变量 FrameNo 标识视频或音频帧编号，成员变量 SliceNo 标识对视频或音频帧进行分组后的分组编号，成员变量 FrameType 标识该分组属于视频帧还是音频帧，成员变量 FrameSize 标识分组所属帧的大小，成员变量 SliceSize 标识分组大小。由于视频数据还需进行 LT 编解码，因此较音频数据携带了一些额外首部信息，包括：成员变量 K 标识分组所属帧中原始分组的个数，成员变

量 K1 标识 LT 编码器输出的编码分组个数，成员变量 ESI 标识编码分组的编号，成员变量 Dgr 标识编码分组对应的度，成员变量 NumA 和 NumB 用于通过线性同余法恢复出编码分组对应的邻居编号。

4.5.2 分组传输

本文采用 Socket 套接字对分组进行传输。Socket 套接字是采用 C 语言的进程间通信的库，其允许不同主机或者同一主机上的不同进程之间进行通信。本文采用了 UDP 协议进行通信，UDP Socket 的通信过程如图 4-8 所示：

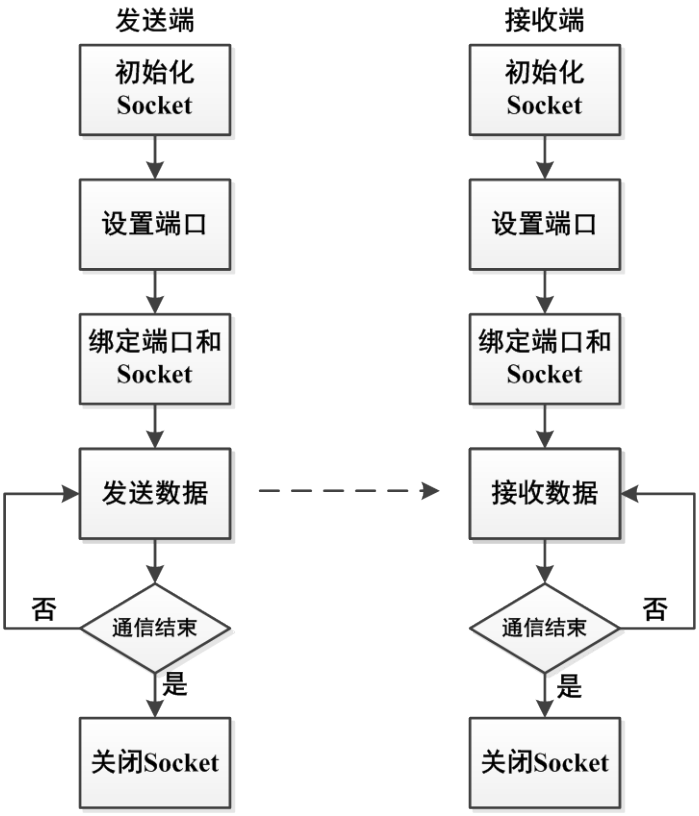

图 4-8 UDP Socket 通信过程

以传输视频数据为例，发送端主要有两项工作：一是利用 Socket 中的 select 功能监听由接收端反馈而来的最优分组长度信息，一旦发现有信息传来则接收该信息并修改输入 LT 编码器的原始分组长度；二是利用 Socket 套接字传输封装后的数据。发送端传输的关键实现代码如下：

```
SocketFd = socket(AF_INET, SOCK_DGRAM, 0);   //初始化 Socket
struct sockaddr_in SAddr;
bzero(&SAddr, sizeof(SAddr));
SAddr.sin_port = htons(SERVER_PORT);
SAddr.sin_family = AF_INET;
```

```
connect(SocketFd, (struct sockaddr *)&SAddr, sizeof(SAddr));
fd_set Inset;   //初始化 select
struct timeval TV;
TV.tv_sec = 0;
TV.tv_usec = 0;
while(True){
    FD_ZERO(&Inset);
    FD_SET(SocketFd, &Inset);
    if(select(SocketFd+1, &Inset, NULL, NULL, &TV)   //利用 select 监听反馈信息
    read(SocketFd, FbData, sizeof(FbData));
    ……
    //发送存放于 OutputBuffer 的数据
    write(SocketFd, OutputBuffer, sizeof(OutputBuffer);
}
```

接收端的主要工作是接收来自发送端的数据的数据并周期性地向接收端反馈最优分组长度信息。接收端实现传输的关键代码如下：

```
struct sockaddr_in ServerAddr, ClientAddr;   //初始化 Socket
SocketFd = socket(AF_INET, SOCK_DGRAM, 0);
ServerAddr.sin_addr.s_addr = htonl(INADDR_ANY);
ServerAddr.sin_port = htons(CLIENT_PORT);
ServerAddr.sin_family = AF_INET;
bind(SocketFd, (struct sockaddr *)&ServerAddr, sizeof(ServerAddr));
RecvBufferSize = 240*1024;   //设置缓冲区大小为 240*1024
RecvOpt = sizeof(RecvBufferSize); //设置缓冲区刷新模式
setsockopt(SocketFd, SOL_SOCKET, SO_RCVBUF, (char *)&RecvBufferSize, RecvOpt);
FlushFb = True;
setsockopt(SocketFd, SOL_SOCKET, SO_KEEPALIVE,
(char *)&FlushFb, sizoef(FlushFb));
setsockopt(SocketFd, IPPROTO_TCP, TCP_NODELAY,
(char *)&FlushFb, sizeof(FlushFb));
while(True){   //接收并反馈数据
    recvfrom(SocketFd, RecvBuffer, sizeof(RecvBuffer), 0, &ServerAddr, &ServerLen);
    ……
    sendto(SocketFd, FbData, FbDataSize, 0, &ServerAddr, sizeof(ServerAddr)); }
```

第五章　系统测试

5.1　视频采集模块的测试

本文为视频采集测试所搭建的测试系统的结构如图 2-1 所示，整个环境由若干个采集节点、若干个中继节点和一个 Sink 节点组成。具体来说，采集节点被分别部署至同一楼层的四个角落处，节点间距为 30-40m。每个采集节点能捕获其所处位置一定范围内的视频和音频数据并发送至相邻的中继节点。经过多个中继节点的自动寻路转发后，由 Sink 节点获得最终数据并进行汇聚处理。之后 Sink 节点将数据转发至与其相连接的 PC 端，由 PC 端对实验结果进行展示。

本文在同一实验环境下进行多次重复测试，以验证多媒体传感节点的稳定性与有效性。具体来说，保持各节点位置不变并选取同一时间段（本文选取 8:30 a.m-11：30 a.m）并进行 30 次重复测试。在 30 次重复测试中，采集模块始终能正确地采集视频数据，采集效果如图 5-1 所示，其稳定性得到验证。另一方面，所采集到的视频数据的分辨率可达 480×272 或 640×480，帧速率可达 20fps-30fps，其有效性得到验证。

图 5-1　视频采集效果

5.2　音频采集模块的测试

音频采集的测试环境与视频采集类似。由于音频数据质量的衡量较为主观，不便于客观呈现，因此本文考虑利用在测试环境下采集一段时长为 60s 的音频数据并封装为 wav 格式，随后采用专业的音频处理软件 Adobe Audition 3.0 观察其各项参数指标。图 5-2、图 5-3 和图 5-4 分别给出了该音频数据的波形图、频谱

图和参数指标。由图 5-4 可以得知，所采集音频数据的采样率达到 8kHz，采样位数达到 16bit，且为单声道，符合课题要求。

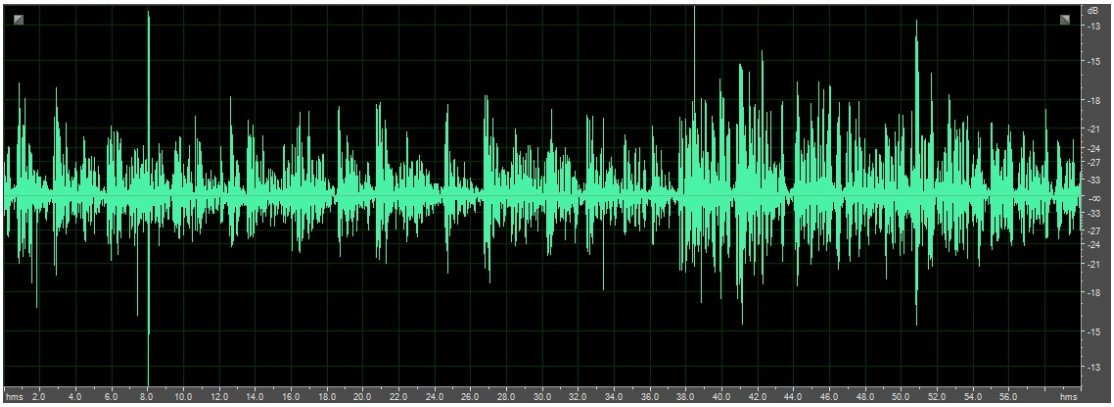

图 5-2　音频数据波形图

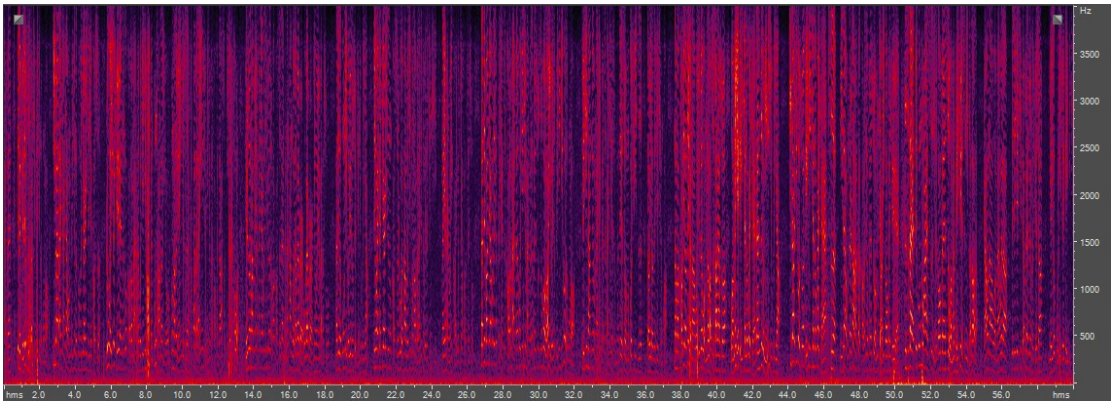

图 5-3　音频数据频谱图

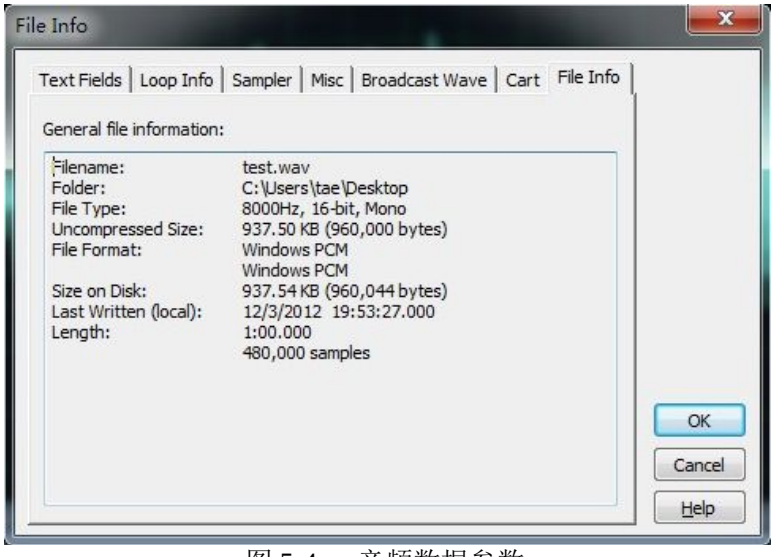

图 5-4　音频数据参数

5.3　OV3640 控制模块的测试

OV3640 控制模块的测试环境是在视频采集测试系统基础上进行了简化，本

文采用两个 S3C6410 嵌入式平台分别作为发送节点和接收节点，二者相距 30-40m，通过无线信道进行通信。另外启用一台 PC 机与接收节点相连，以展示测试结果。测试方案为：首先启动视频采集模块进行实时的视频数据采集，然后启动 OV3640 控制模块，对 OV3640 视频传感器进行各项控制，并在 PC 端观察控制效果。图 5-5 和图 5-6 分别给出了数码变焦、镜像以及翻转的测试效果。

图 5-5　数码变焦测试效果

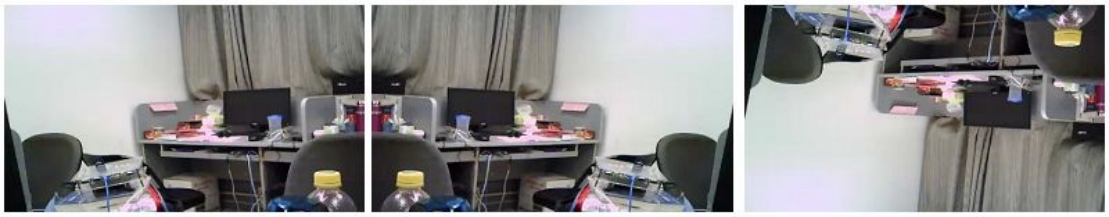

图 5-6　镜像和翻转测试效果

5.4　LT 编解码模块的测试

LT 编解码模块的测试环境与前面几节有所区别，搭建该模块的测试环境时对网络规模、节点部署位置、节点持续工作时间等方面不作过多考虑，转而采用较为简单的测试环境，即只采用两个 S3C6410 嵌入式平台分别作为发送节点和接收节点，二者相距 30-40m 并通过无线信道进行通信，另外启用一台 PC 机与接收节点相连以展示测试结果。这样做的目的在于尽可能地排除其他影响视频传输质量的可能因素，以提高测试结果的可靠性。LT 编解码模块测试环境的搭建关键在于需要确保测试环境始终不变，包括节点的部署，测试时间点和测试时长的选取，网络环境（即信道丢包率），传输对象等等。其中信道丢包率是非常重要的测试环境因素之一，直接影响最终测试结果。为确保信道丢包率的可控性，本文采用通过 Gilbert 模型逆向实现的信道丢包率模拟模块，该模块的原理和实现细节不在本文讨论范围内，故不再赘述，只需知道利用该模块能够灵活控制信道丢包率。

本文对 LT 编解码模块采用的测试方案为：在相同测试环境下对比启用 LT 编解码子模块与未启用 LT 编解码子模块两种情况下接收端收到视频数据的 PSNR 曲线[23]。PSNR 的主要思想是将视频流中的每一帧的每一个像素与参考视频（一般是原始视频）中的对应帧的对应像素进行比较，PSNR 值越高则说明被

比较视频越接近原始视频，传输质量相应地也越好。若启用 LT 编解码子模块的 PSNR 曲线优于未启用情况下的，则 LT 编解码子模块的有效性就能够得到验证。图 5-7 和图 5-8 分别给出了丢包率 P_{lost}=10%时未采用 LT 模块和采用 LT 模块后的 PSNR 曲线。

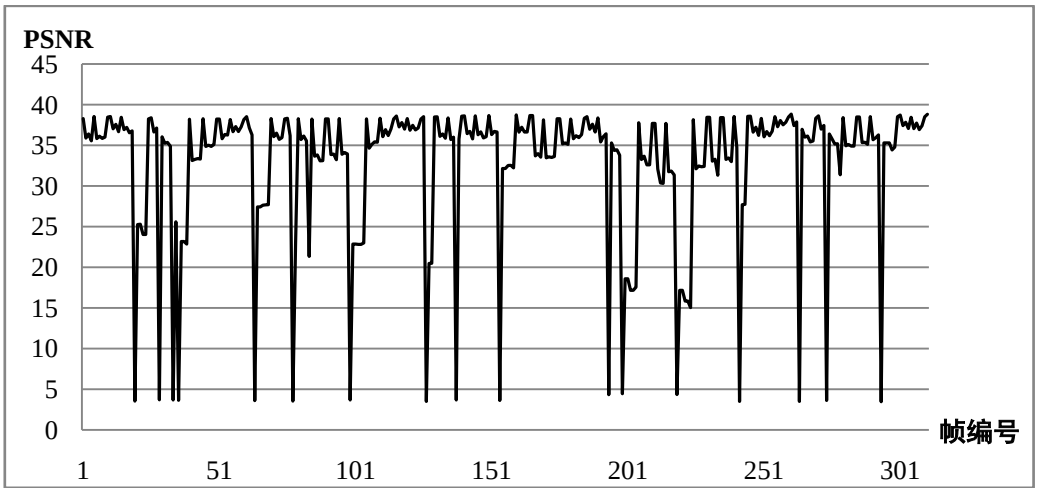

图 5-7　未采用 LT 的 PSNR 曲线

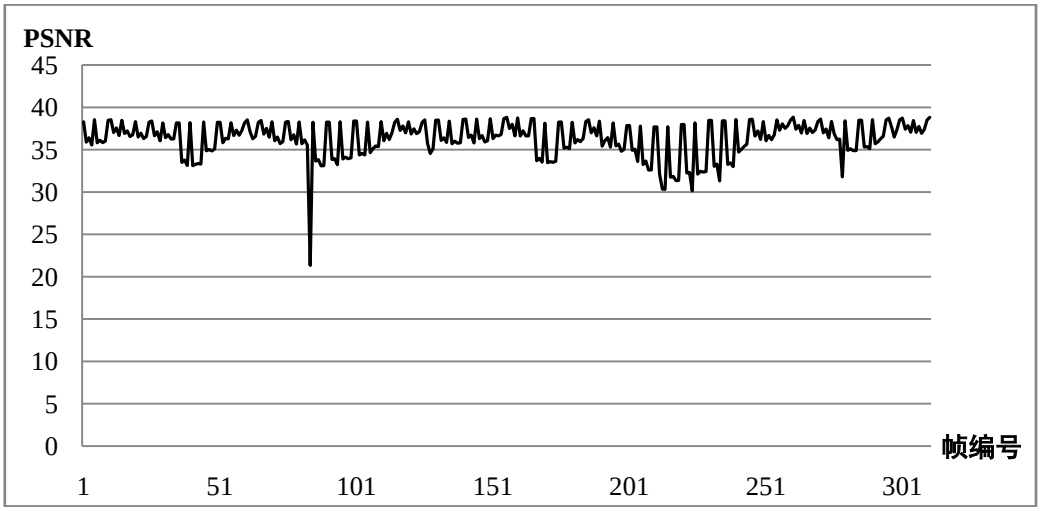

图 5-8　采用 LT 后的 PSNR 曲线

从图 5-7 和图 5-8 的对比中可以看出，在丢包率 P_{lost}=10%情况下，未采用 LT 编解码模块的 PSNR 曲线中某些点的 PSNR 值衰落至很低的水平，显示出此处有丢帧现象。而在采用了 LT 编解码模块的 PSNR 曲线中很少出现 PSNR 值衰落的现象，绝大部分丢帧现象被克服了。由此可以说明 LT 编解码模块能在一定程度上提高系统的抗干扰性。

第六章　总结与展望

6.1　论文工作总结

截止目前为止，本文基本完成了对 WMSNs 中采集模块和传输模块的研究与实现，并进行了一系列的测试工作。本文首先分析了 WSNs 和 WMSNs 的研究现状和发展前景，并在此基础上提出了 WMSNs 中采集模块和传输模块的一种解决方案。接着本文对上述两个模块的设计实现进行了深入的阐述，包括研究过程中涉及到的相关技术，理论原理，各模块的系统框图以及部分相关伪码等。最后本文进行了一系列的测试工作，以检验前述各项工作的完成情况。

实验测试证明，本文所实现的采集模块和传输模块均能达到课题要求。采集模块能够长时间稳定地工作并提供高质量的视频数据和音频数据，同时能对 OV3640 视频传感器进行灵活地控制。传输模块能够稳定地进行 LT 编码工作，LT 编码的应用在一定程度上能够提高系统的抗干扰性，在较差网络环境下避免 PSNR 曲线频繁衰落，直观来看即减少丢帧现象的发生。同时传输模块还实现了最优分组长度实时反馈机制，该机制能够在保证 LT 编码质量的同时有效地降低 LT 编码带来的数据冗余。

6.2　改进建议

虽然系统各模块已达到预期要求，但目前仍存在以下几项待改进之处：

（1）OV3640 视频传感器控制功能有待丰富。在实际应用中可能需要对 OV3640 进行更多地控制，譬如在不同光照环境下手动调整曝光和白平衡，视频裁剪，输出静态图像等等。后续工作中可以参照 OV3640 相关手册查询上述功能对应的寄存器地址及其修改方法。

（2）多描述编码功能有待实现。本文虽采用了 S3C6410 嵌入式开发平台内部进程的多媒体编码器 MFC 进行 H.264 硬编码，但并未发挥出其全部功效。实际上 S3C6410 可同时开启多个 H.264 硬件编码器，利用此特性可实现对视频数据的多描述编码功能。相较于单描述编码，多描述编码的可靠性更佳，最终接收到的视频图像质量更好。

（3）音频数据编码有待实现。本文之所以未对采集到的音频数据进行信源编码，是考虑到相较于视频数据，音频数据的数据量较小，对带宽的占用比例较低，若耗费较为紧张的 CPU 资源进行编码可能会造成系统的实时性下降。在后续工作中，若能找到计算能力更强的硬件平台或针对 ARM 平台优化更好的音频编码技术，则可考虑将该编码器引入到系统中以压缩音频数据。

参考文献

[1] 王汝传, 孙力娟. WMSNs 技术[M]. 北京: 人民邮电出版社, 2011.

[2] 卢娟. WMSNs 研究初探[J]. 大众科技, 2012, 14(4): 73-74.

[3] Gregory J.Pottie, William J.Kaiser. Wireless Integrated Network Sensors[J]. Communications of the ACM, 2000, 43(5): 51-58.

[4] A.P. Chandrakasan. Power-aware Wireless MicrLinuxensor Networks[R]. DARPA PAC/C Meeting, 2000-05-23.

[5] 马华东, 陶丹. 多媒体传感器网络及其进展[J]. 软件学报, 2006, 17(9): 2013-2018.

[6] Feng W, Code B, et al. Panoptes: A Scalable Architecture for Video Sensor Networking Applications[A]. In: Rowe L, Vin H, eds. Proc. of the ACM Int'l Conf. on Multimedia 2003. New York: ACM Press, 2003. 151-167.

[7] Kulkarni P, Ganesan D, Shenoy P, Lu QF. SensEye: A Multi-tier Camera Sensor Network[A]. In: Zhang HJ, Chua T-S, eds. Proc. of the 13th Annual ACM International Conference on Multimedia'05. New York: ACM Press, 2005. 229-238.

[8] Luby M. LT Codes[C]. Proceedings of the 43rd Annual IEEE SympLinuxium on the Foundations of Computer Science (STOC), Vancouver, Canada, November, 2002: 271-280.

[9] 于明, 范书瑞, 曾祥烨. ARM9 嵌入式系统设计与开发教程[M]. 北京: 电子工业出版社, 2006.

[10] 王国伟, 曾碧,谭昌盛. 基于 S3C6410 的 H.264 视频采集传输系统[J]. 微计算机信息, 2011, 27(5): 106-108.

[11] OmniVision Technologies, Inc. OV3640 3.1-Megapixel Product Brief[OL]. [2007-12]. http://www.ovt.com/uploads/parts/OV3640_PB(1.02)_web.pdf.

[12] 王旭阳. 基于 ARM 的无线视频监控系统的设计与实现[D]. 北京: 北京邮电大学, 2012.

[13] 齐文钊. 基于 FPGA 的视频编码器设计[D]. 天津: 天津大学, 2004.

[14] 毕厚杰, 王健. 新一代视频压缩编码标准 H.264[M]. 北京: 人民邮电出版社, 2009.

[15] 岳殿武. 分组编码学[M]. 西安: 西安电子科技大学出版社, 2007.

[16] 张荧俊. 擦除码在 WSNs 可靠数据传输中的应用[D]. 西安: 西安电子科技大学, 2010.

[17] 朱宏鹏, 张更新, 谢智东. 喷泉码中 LT 码的次优度分布[J]. 应用科学学报, 2009, 27(1): 6-11.

[18] 李亮, 赵加祥, 袁鑫. 基于 NSD 度分布函数的 LT 码构造[J]. 计算机工程, 2010, 36(15): 240-244.

[19] Chris Harrelson, Lawrence Ip, Wei Wang. Limited Randomness LT Codes[C/OL]. Proceedings of 41st Annual Allerton Conference on Communication, Control, and Computing, Monticello, Illinois, October, 2003. http://www.eecs.berkley.edu.

[20] Gilbert E N. Capacity of A Burst-noise Channel[J]. The Bell System Technical Journal, 1960, 39(9): 1253.

[21] Sanneck H, Carle G. A Framework Model for Packet Loss Metrics Based on Loss Runlengths[A]. Proceedings of the SPIE/ACM SIGMM Multimedia Computing and Networking Conference 2000(MMCN 2000)[C], CA(USA): San Jose, 2000: 177-187.

[22] Jiang W, Schulzrinne H. Modeling of Packet Loss and Delay and Their Effect on Real-time Multimedia Service Quality[A]. Proceedings of NOSSDAV2000[C], NC: Chapel Hill, 2000.

[23] An Chan, Kai Zeng, Prasant Mohapatra, Sung-Ju Lee and Sujata Banerjee. Metrics for Evaluating Video Streaming Quality in Lossy IEEE 802.11 Wireless Networks[A]. IEEE Conference on Communications(INFOCOM), Mar. 2010.

致谢

光阴荏苒，岁月如梭，转眼两载有余。我即将收拾行囊，挥别往昔，踏上新的旅程。回首研究生生涯，往事历历在目。在这里，我度过了人生中最忙碌也是最充实的一段时光。在这里，我为母校浓厚的学术氛围所感染。也是在这里，我对立身处世之道有了更深刻的认识。这些收获无不得益于师长与同窗的无私帮助，借本文即将完成之际我要向他们致以最真挚的谢意。

首先要深深地感谢我的导师李文生老师。在论文的撰写过程中我得到了李老师的悉心指导，她严谨的治学态度与渊博的学识让我受益匪浅，我的点滴成长和进步无不包含着老师的淳淳教诲。在此由衷地感谢李老师，祝她身体健康，工作顺利。

感谢段鹏瑞老师。初入实验室之时段老师的耐心指导使我迅速成长，项目进展过程中与段老师反复的讨论交流也令我获益良多。感谢段老师让我懂得了什么是执着的科研精神。

感谢王旭阳师兄。在项目中遭遇技术难关时常会得到师兄的耐心指点，我在实验室项目中所取得的成果与师兄的无私帮助不可分离。

感谢郭奇、李强、张默、杨天昊、刘梦轩和左佳。他们让我体会到了团队合作的力量与互助的快乐。

最后，我要感谢我的父母，他们给予我的鼓励和关心是我前进的最大动力。

攻读学位期间发表的学术论文

[1] 钱乐, 李文生. 基于 S3C6410 的多媒体传感节点的研究与实践. 中国科技论文在线, 2012.