ABSTRACT

Image Transmission is the technology which is used to send the images from distant place to local place through the communication network—Internet or Wireless network in order to share the video signals. It is the core of the multimedia communication technology, and can be used in many typical multimedia communication operations such as video meeting system and safety monitor system.

To realize the low bitrate and high quality image transmission is the objective of many communication service providers and research institutes. We should not only satisfy the clients' high standard demand, but also overcome the bottle-neck of bandwidth of network, save the resource of the communication line, and reduce the cost of image transmission.

This dissertation gives an example of video monitor system, which is used to research the technology of image transmission. The example introduced the design and realization of an image transmission application based on H.264 encode/decode technology. In general, the system provides two transimission plans, one uses Internet, and the other one uses GPRS network. In image collecting terminal, Hikvision video-encode card is used to compress video signal to H.264 format and transmit it by TCP/IP and UDP/IP socket or GPRS module. In receiving end, the system decodes the H.264 bitstream for display.

For its low bitrate character, this system can be applied in normal 10M-LAN and GPRS service fields. And we can also develop our special image transmission system on the basis of it.

Key Words: Image Tranmission, Video Encode/Decode, H.264, GPRS, Visual C++

学位论文版权使用授权书

本人完全了解同济大学关于收集、保存、使用学位论文的规定，同意如下各项内容：按照学校要求提交学位论文的印刷本和电子版本；学校有权保存学位论文的印刷本和电子版，并采用影印、缩印、扫描、数字化或其它手段保存论文；学校有权提供目录检索以及提供本学位论文全文或者部分的阅览服务；学校有权按有关规定向国家有关部门或者机构送交论文的复印件和电子版；在不以赢利为目的的前提下，学校可以适当复制论文的部分或全部内容用于学术活动。

学位论文作者签名：杨文波

2007 年 3 月 7 日

经指导教师同意，本学位论文属于保密，在　　年解密后适用本授权书。

指导教师签名：　　学位论文作者签名：

年　月　日　　年　月　日

同济大学学位论文原创性声明

本人郑重声明：所呈交的学位论文，是本人在导师指导下，进行研究工作所取得的成果。除文中已经注明引用的内容外，本学位论文的研究成果不包含任何他人创作的、已公开发表或者没有公开发表的作品的内容。对本论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确方式标明。本学位论文原创性声明的法律责任由本人承担。

签名：杨文波

2007年 3月 7日

第 1 章 绪论

1.1 图像传输技术的现状和发展前景

图像传输是现代社会每时每刻都在进行的工作，已经成为社会日常生活中必不可少的信息产品，从电视、电影、会议电视，到视频监控、可视电话等，无处不有它的影子。

人类在接受信息的来源方面，靠眼睛摄入的图像信息占 80%以上。在现代社会，人们迫切需要了解全球正在发生的社会事件，而且人们不满足于文字报道、语音报道，更需要从现场视频图像来看到真实直观的场面。如何实时、高效、可靠、简便地将视频信号传送到目的地，成为许多科研人员和相关设备生产厂商研究的课题。因此，图像传输设备有着巨大的市场需求。

本文主要以视频监控系统为实例，对实时图像的传输技术进行研究。

1.1.1 当前图像传输技术概述

随着计算机技术的完善普及，INTERNET 和无线通讯网络广泛普及，高速宽带主干网的建成以及高速接入网的迅速发展，再加上多媒体技术、视频压缩编码技术、网络通讯技术的发展，图像传输技术步入了一个更高的境界，即数字化、网络化、模块化。

就视频监控系统来讲，现今主要有两种类型，一种是以数字录像设备为核心的视频监控系统，另一种是以 WEB 服务器为核心的视频监控系统。前者是基于嵌入式的数字系统，通过高像素摄像头采集视频，然后高精度高速 AD 转换器将采集到的视频转化为数字图像，再由视频压缩专用芯片将数字图像压缩编码，将压缩视频存储于嵌入式设备自带的存储器中，便于事后回放。后者的视频采集及压缩方法与前者基本相同，它对压缩视频的处理不只是局限于本地存储，而且通过网络将视频信号发送给客户端，系统基本能做到实时传输，快速反应。

为了实现远程观看现场图像的要求，关键是解决图像实时传输问题，而图像的信息量巨大，在传输过程中需要很大的线路带宽资源。模拟电视图像占 8MHz 带宽，数字电视原始图像需要 170Mbps 的传输码率。但是宽带远距离通

信线路的使用费十分昂贵。因此，为了降低图像传输的费用，必须压缩其占用的线路带宽，即采用图像信息压缩技术；而图像压缩在模拟信号方式下无法实现，只能在数字信号方式下进行。

目前有 MPEG 和 H.两大类压缩标准。对于图像传输的场合，需要较低的码率来保证传输的实时性，通常采用的压缩格式是 MPEG-4 和 H.264，两者都能在低码率前提下提供较好的图像质量，很适合网络图像传输。采用当前的高速嵌入式系统及相关算法，实时地将视频图像压缩为这两种格式已不是难题。

网络传输速度是图像传输关键因素。就目前的传输途径来讲，基本都是基于 INTERNET 的，因为 INTERNET 的网络速度能满足 MPEG-4 和 H.264 两种视频格式的传输，从而满足图像传输实时性的要求。但随着无线通讯网络的日益完善以及 3G 网络的应用，高质量图像的无线实时传输也指日可待。

1.1.2 影响图像传输技术发展的因素

图像传输技术发展的同时，有些因素也在制约或影响着它的发展。这些因素很多，归纳起来主要表现为以下几点：

（1）视频压缩技术

网络视频监控技术发展的关键是视频压缩技术。目前在网络视频监控领域应用到的视频压缩技术主要是 M-JPEG, MPGE-4 与 H.264。M-JPEG 压缩技术，主要是基于静态视频压缩发展起来的技术，基本不考虑视频流中不同帧之间的变化，只单独对某一帧进行压缩。优势是可以获取清晰度很高的视频图像，并可灵活设置每路视频清晰度、压缩帧数；劣势是受处理速度所限，无法完成实时压缩，丢帧现象比较严重，单帧视频占用空间较大。近两年，MPEG-4 压缩技术的发展一直倍受业界关注，2001 年年初由 MPEG 专家组公布的 MPEG-4 V2.0 版本已经正式成为国际标准。这种压缩方式保有极佳的音质和画质，压缩率可超过 100 倍；并且满足了低码率应用的需求，可利用较少的数据，获取最佳的图像质量。更适合于远程图像传输。由 ITU-T 和 MPEG 联合开发的新标准 H.264，引入了面向 IP 包的编码机制，有利于网络中的分组传输，支持网络中视频的流媒体传输，比 H.263 和 MPEG-4 节约了 50% 的码率，具有更高的压缩比和更好的信道适应性。

除以上几种技术外，还处于探索阶段的，比较受业界关注的视频压缩技术

是 MPEG-7 与 MPEG-21。 MPEG-7 可对大量的图像、声音信息进行管理和迅速搜索。MPEG-21 由 MPEG-7 发展而来，据有关消息，MPEG-21 主要规定数字节目的网上实时交换协议。

就目前来说，2006 年网络视频监控产品的压缩方式主要表现为 MPEG-4 与 H.264，虽然网络视频监控企业在不断的寻求更好的视频压缩传输方式，但到目前为止，网络视频传输的实时性仍不是太令人满意，用户期待更好的产品出现。

（2）网络普及程度和网络传输速度

压缩技术的发展是影响图像传输技术发展的关键因素。除此之外，由于网络远程传输主要通过局域网、广域网、互联网以及无线网络来实现音视频传输，所以网络的普及程度和网络传输速度对图像传输技术的发展也很重要。目前我国图像传输技术发展的基础条件在不断提高，随着网络的普及、带宽的提高，网络图像传输的应用前景也越来越广。

（3）解决方案及相关投入

图像传输过程中所涉及到的两个关键点——图像压缩和图像传送，两者的实现有着不同的方案。图像压缩可分为硬压缩和软压缩，硬压缩是指采用专用的高速嵌入式压缩设备压缩视频，软压缩则是通过压缩软件、利用 PC 机处理器压缩视频。硬压缩技术不占用控制系统资源，实时性好，但投入较大，一块专用的压缩板卡售价在千元以上。软压缩应用方便，实时性、投入小、可靠性较低，适合家庭娱乐采用。而图像传输网络又分为公网和专网、有线网络和无线网络。根据各自的特点，方案投入也相差很大。

我国市场对价格一向比较敏感，虽然随着技术的发展进步，有些视频设备已调低了价格，但某种方案的广泛应用，也会影响到其相关技术的发展。

1.1.3 主要相关产品的现状及发展趋势

图像传输技术的典型应用即是视频监控系统。在国内外市场上，主要推出的是数字控制的模拟视频监控和数字视频监控两类产品。前者技术发展已经非常成熟、性能稳定，并在实际工程应用中得到广泛应用，特别是在大、中型视频监控工程中的应用尤为广泛；后者是新近崛起的以计算机技术及图像视频压缩为核心的新型视频监控系统，该系统解决了模拟系统部分弊端而迅速崛起，但仍需进一步完善和发展。目前，视频监控系统正处在数控模拟系统与数字系

统混合应用并将逐渐向全数字系统过渡的阶段。嵌入式方式的视频监控系统主要是以嵌入式视频 WEB 服务器方式提供视频监控。其具有布控区域广阔、几乎无限的无缝扩展能力、易于组成网络视频监控系统的设计与实现非常复杂的监控网络、性能稳定可靠等特点，必将成为今后视频监控领域的主流产品。

1.2 课题研究目的、理论意义和实际应用价值

为了提高工作生活的安全性和舒适性，视频监视系统应用非常广泛，从道路路口到企业厂，再到交通工具，无处不有它的影子。视频监视系统的应用，在一定程度上减少了事故的发生。但随着其应用日益广泛，对视频监视系统的要求也越来越高，不单局限于事后“有据可查”，还要能做到“快速反应，及时应对”。固定的监视设备较容易满足这一要求，而对于移动设施，如火车、汽车等，在图像传输技术上还存在难点。

“图像传输技术的研究”主要目的是实时、有效、低成本地将视频图像传回到终端。涉及到的关键技术有：图像编解码技术、INTERNET 图像传输技术、无线图像传输技术、嵌入式软硬件设计以及 PC 软件设计技术等等。

视频监控系统应用日益广泛，社会各部门、各行业及居民小区纷纷建立起了各自独立的监控系统或报警系统。建立和不断完善的安防系统，对保护人员和设备安全、提高生产和管理效率、预防和制止犯罪、维护社会经济稳定起到了重要作用，但广泛的应用对视频监视系统的要求也越来越高。因此，研究高效率的图像传输技术具有较深的现实意义。

1.3 研究方法、技术路线、实施方案

本文以一个小型的视频监控系统为实例，来研究图像传输技术，本课题的关键技术包括视频的压缩编码和视频数据的网络传输。

在服务器端，系统采用海康公司的 DS-4004HC 视频采集压缩卡压缩视音频，以 Visual C++为程序开发平台，对视音频流进行处理，实现本地回放和网络发送。在客户端，同样以 Visual C++ 平台开发回放软件，存储、回放服务器端发送过来的视频信息。图像传输网络采用有线和无线两种，以适用于不同的应用场合。系统主要框图如下：

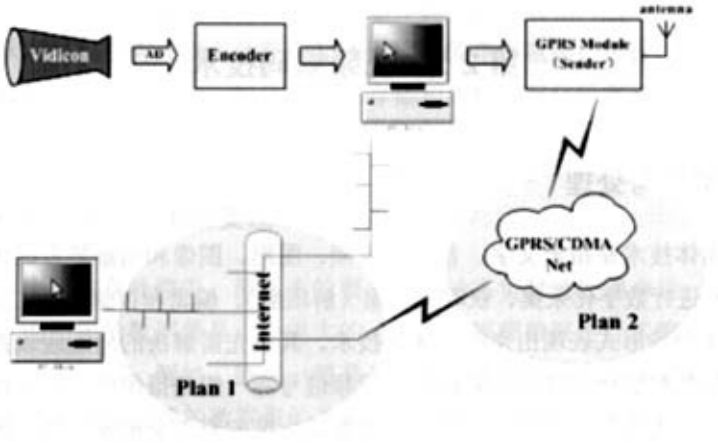


图 1.1 视频监控系统框图

（1）图像采集及格式转换：高像素摄像头采集视频，DS-4004HC 视频采集压缩卡通过高精度高速 AD 转换器将采集到的视频转化为数字图像，再由板卡上的视频压缩专用芯片将数字图像压缩编码为 H.264 图像格式供系统传输。

（2）图像数据发送：以 Visual C++为开发平台，设计视频图像的本地处理程序。该程序用于读取视频采集卡的视频数据，一方面在本地服务器上实时播放视频图像，另一方面实时地将视频数据通过网络传送给客户端。视频信号的传输方案有两种：①因为传输的图像信号的数据量很大，故选用效率较高的 UDP 协议传输视频信号。服务器端应用程序从视频卡处读取的视频数据直接通过 INTERNET 发送给客户端，该方案可以提供给客户端质量较高的视频图像。②服务器端应用程序将视频数据发送到串口，串口处外接嵌入式 GPRS 通讯设备，该设备采用高速 MCU 为控制芯片，实时读取串口数据并将其通过 GPRS 模块无线发送图像数据。由于无线网络的数据传输能力有限，该方案提供的视频图像质量较低。

（3）图像接收回放：在客户端开发以 Visual C++为平台的视频图像接收回放程序。该程序直接连接到 INTERNET，通过客户端 IP 地址获取视频数据。由于 GPRS 网络也是接入互联网的，所以对于每一个 GPRS 用户，网络都会分配给它一个 IP 地址。因此，对于两种图像传输方案，客户端都可以采用同样的方式获取视频数据流，并且回放、存储。

第 2 章 视频编码技术

2.1 视频信号处理

多媒体技术是指把文字、音频、视频、图形、图像和动画等多媒体信息通过计算机进行数字化采集、获取、压缩（解压缩）、编辑和存储等加工处理，在以单独或合成形式表现出来的一体化技术。其首先需解决的问题应该是把声音和视频信息数字化后送到计算机中。视频信号源一般是摄像机、录音机、扫描仪以及视频光盘等，而它们的输出大多数是标准的彩色全电视信号，因此视频信号的获取主要是标准的彩色全电视信号的获取。

目前通用的电视标准有 NTSC 制式、PAL 制式和 SECAM 制式三种。虽然这三种制式的信号处理方式不同（如彩色空间、相位处理各不相同），但其视频获取的流程却是一样的，即需要首先将彩色全电视信号经过采集设备分解成模拟的 RGB 信号或 YUV 信号，然后进行各个分量的 A/D 变换、解码，将模拟的 RGB 或 YUV 信号转换成数字的 RGB 信号或 YUV 信号，存入帧存储器，主机可通过总线对帧存储器中的图像数据进行处理，帧存储器中的数字 RGB 信号或 YUV 信号经过 D/A 变化转换成模拟的 RGB 或 YUV 信号，再经编码合成彩色全电视信号，输出到显示器上。其视频处理流程如下图所示：

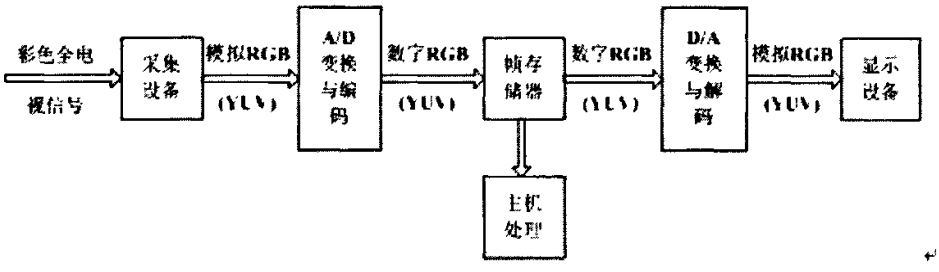


图 2.1 视频获取流程

2.2 视频压缩的必要性和发展前景

多媒体信息经数字化处理后具有加密、抗干扰能力强、可再生中继等优点，

但同时也伴随海量数据的产生，这对信息存储设备及通讯网络均提出了很高的要求，从而成为阻碍人们有效获取和使用信息的重大瓶颈。因此，研究高效的多媒体数据压缩编码方法，以压缩形式存储和传输数字化的多媒体信息具有重要的意义。

2.2.1 视频压缩编码的必要性

在多媒体信息处理中，最基本的要求是能动态实时地处理声音、动画、视频信号，而图像的数据量是十分庞大的，若不对视频数据进行压缩处理，实时性根本就不能达到。例如，针对一幅具有中等分辨率（640×480）的彩色（24 bits/pixel）数字视频图像的数据量约 7.37 Mbits 帧，帧速率 30 帧秒（NTSC 制式），则视频信号的传送速率大约为 221.1 Mbits/s，一分钟的视频表演则需要 13266 M 以上的硬盘空间。数据量大，难以寻求庞大的存储设备存储这些数据，而且计算机也难以实时地从存储器将这些数据传送到中央处理器，因此，视频数据压缩技术也就成了开发多媒体系统中视频处理的关键技术。

通过对视频数据的分析发现，原始视频数据存在的冗余度为数据压缩的实现提供了可能。首先，对于每帧图像数据存在很大的空间冗余，视频图像帧内临近像素之间是空域相关的。其次，对于由每秒 30 帧组成的视频序列信号，其相继帧之间也具有较强的相关性，即存在时间冗余。例如，对于电视中的演讲人图像序列，相邻帧之间可能只有由头部、眼部、嘴部的微小变动而引起的细小差别。再次，因为在多媒体系统的应用领域中，人是主要接收者，眼睛是图像信息的接收端，这样就有可能利用人的视觉对于边缘急剧变化不敏感（视觉掩盖效应）和眼睛对图像的亮度信息敏感，对颜色分辨率弱的特点实现高压缩率，从而使有压缩数据恢复的图像信号仍有满意的主观质量。

一旦优秀的视频压缩技术投入使用，使得低成本、较低速率要求、有限带宽的条件下得以动态实时地处理高质量的运动图像，它将对社会产生深远的影响并具有广阔的应用领域。多媒体与互联网相结合，使得多媒体通信系统能提供可视通信、远程监控、远程教学、机中图像管理和声像资料联网传输等功能，利用多媒体计算机系统进行教育、训练、演示、咨询、家庭娱乐等，将常规电视数字化及制造高清晰度电视（DHTV）、交互式电视系统等，而当前多媒体市场的繁荣也证实这种发展前景正一步步向人们走近。

2.2.2 视频编码技术的发展

视频编码是数字图像处理中的一个重要的研究领域。视频编码的一个主要目的是在保证一定重构质量的前提下，以尽量少的比特数表征视频信息。

传统的视频编码方式是以视频信号的数字量为编码对象的，用统计概率模型来描述信源，编码实体是像素或像素块，与视频信息的内容无关。它将整个视频信号作为一个内容单体来处理，其本身不可再分割，而这与人类对视觉信息的判别法则，也就是大脑对视神经导入的视觉信号的处理方法是完全不同的。这就决定了不可能将一个视频信息完整的从视频喜好中提取出来，比如，将加有台标和字幕的视频恢复成无台标、字幕的视频。解决问题的唯一途径就是在编码时就将不同的视频信息载体——视频对象 VO（Video Objects）区分开，独立编码传送，将图像序列中的每一帧，看成是由不同的 VO 加上活动的背景所组成。VO 可以是人或物，也可以是计算机生成的 2D 或者 3D 图形。VO 具有音频属性，其属性赋值可能是“有”或者是“无”。但音频的具体内容数据是独立于视频编码、传输的。VO 概念的引入，更加符合人脑对视觉信息的处理方式，并使视频信号的处理方式从数字化进展到智能化。提高视频信号的交互性和灵活性，使得更广泛的视频应用和更多的内容交互功能成为可能。

现代图像编码理论指出，人眼捕获图像信息的本质是“轮廓——纹理”，使人眼感兴趣的是 VO 的一些表面特性，如形状、运动和纹理等。VO 的表面往往是不规则的、千变万化的，但可将其视为一定视角下，n 个形状规则的、具有一定纹理的剖面的组合和连续运动，这些剖面的组合称为视频对象面 VOP（Video Object Plane）。VOP 描述了 VO 在一定视角条件下的表面特性。VOP 的编码主要由两部分组成：一个是形状编码，另一个是纹理和运动信息编码。假设输入的视频序列的每一帧都被分割成多个任意形状的图像区域（图像对象面），每个区域可能覆盖场景中特定的感兴趣的图像或视频内容。

由此可见，第一代视频编码技术并未考虑信息接受者的主观特性、视频信息的具体含义和重要程度等，只是力图去除数据的冗余，这是一种低层次的编码技术。真正代表视频编码方向的是基于内容的第二代视频编码技术，它所关心的是如何去除视频内容的冗余。它认为人眼是视频信号的最终接收者，在视频编码时应充分考虑人眼视觉特性的影响，这是目前视频编码最为活跃的一个领域。H.264 和 MPEG-4 标准采用的就是基于内容的第二代视频编码技术。

2.3 常用的视频压缩技术

数据压缩的分类方法繁多，在这里对分类方法等问题不进行讨论，只讨论在视频压缩技术中涉及到的一些常用数据压缩技术。

2.3.1 预测编码

预测编码是基于统计冗余数据压缩理论的一种编码方法。它是按某一模型利用以往的样本值对新样本进行预测，然后将样本中的实际值与其预测值相减得到一个误差值，并对这一误差值编码。由于误差值远远小于实际值，从而达到压缩数据的目的。预测编码是一种无损压缩。

预测编码有线性预测和非线性预测两大类。线性预测编码又称为差分脉冲编码调制，即 DPCM（differential pulse code modulation）方法。

其工作原理为：假定已有样本序列 X_1、X_2.. ..X_{n-1}，预测值为 X_n'，实际值为 X_n，a_i (i=1，2，…，n-1)为系数，e_n 为误差值，则：

$$X_n' = \sum_{i=1}^{n-1} a_i X_i \tag{2.1}$$

$$e_n = X_n - X_n' \tag{2.2}$$

最优线性预测就是选择预测系数 a_i，使 e_n 的均方值最小。

帧内预测编码一般采用像素预测形式的 DPCM，其优点是算法简单，易于硬件实现。缺点是对信道噪声和误码很敏感，会产生误码扩散，使图像质量大大下降。帧内 DPCM 的编码压缩比很低，现在已很少独立使用，一般要结合其它编码方法。

帧间编码技术处理的对象是序列图像，它是把几帧的图像存储起来做实时处理，利用帧间的时间相关性进一步消除图像信号的冗余度，提高压缩比，其技术基础是帧间预测技术。

帧间预测编码主要利用视频序列相邻帧间的相关性，即图像数据的时间冗余来达到压缩的目的，可以获得比帧内预测编码高得多的压缩比。帧间预测编码一般是针对图像块的预测编码，主要方法有帧重复法、帧内插法、运动补偿法等。其中运动补偿预测编码效果最好，已为各种图像视频编码标准所采用。

运动补偿方法是跟踪画面内的运动情况对其加以补偿之后再进行帧间预

测，它能较好地提高压缩比，得到了特别的重视和广泛的应用。

运动补偿预测技术通常由以下几个方面组成：

① 首先把图像分割为静止和运动的两部分，这里假设运动物体仅作平移；

② 估计物体的位移值；

③ 用位移估值（即运动矢量）进行运动补偿预测；

④ 预测信息编码。

在这里，图像分割是运动补偿预测的基础，实际上把图像分割成不同运动的物体比较困难，从而通常采用两种比较简单的方法：一种是把图像分为矩形子块，适当选择块的大小，把子块分为动和不动两种，估计出运动子块的位移，进行预测传输；另一种方法是对每个像素的位移进行递归估计。

在图像分割的基础上，对运动子块的估计（或者说运动估值 ME）便成了运动补偿预测的关键技术，一旦求得运动物体的运动矢量后，即可将其送入 MC 预测器（如图 2.2 所示），进行编码传输。

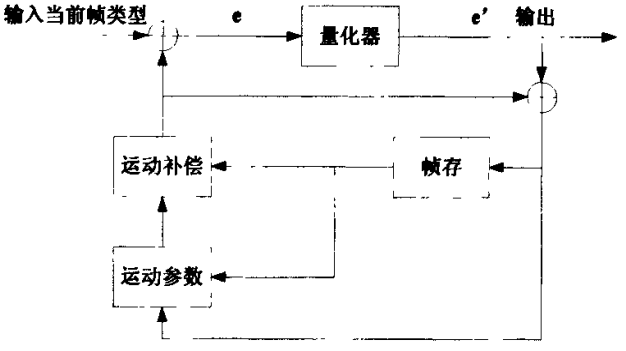


图 2.2 运动补偿预测编码器

由于在运动图像编码中较多地关心由被摄物体和摄像机二者之间的运动共同造成的物体图像的二维运动，高效而简捷的运动矢量求取算法就成了运动补偿技术中的焦点问题。目前常用的几种对运动矢量进行估值的方法有：

（1）块匹配算法：BMA（Block Matching Algorithm）是目前常用的运动估计算法，它假设块内像素只做相等的平移。在该算法中要求确定块尺寸大小的选择，只有在块很小时才可近似认为块内各点做相等的平移，以满足 BMA 的基本假设。但若块太小，则估计结果易受干扰噪声影响不够可靠，且传送运动矢量所需比特数过多；块取大则可减轻其影响，但 BMA 法的基本假设难以满足，

影响估计精度，且大块中常包含多个不同运动的物体，块内运动一致性更难满足。作为折中考虑，MPEG-2 标准选取 16×16 像素点阵作为块匹配单元。

（2）像素递归法：它能适应运动补偿内插中对每个像素的运动进行精确到亚像元级的估计要求，由于每个像素都有一个对应的运动矢量，为了降低码率而避免将其所有的运动矢量都进行传输，提出了许多解决的办法。如：让接收端在与发射端相同的条件下用与发射端相同的方法进行运动估计、Netravali 像素递归法的迭代修正，块递归和迭代修正相结合的运动估计等。

（3）相位相关法：由于物体的空间位移与其相位变化相对应，从而可将运动估计转至频率域进行，这其中需要进行傅立叶变换求得相位相关函数，利用相位相关函数的尖峰求取运动矢量估值。

在当前的视频压缩中运动补偿技术是关键，而运动矢量的求取又是关键中的关键，运动估计的实时性和估计精度是一对基本矛盾。在现时的技术条件下及常见应用中，块匹配算法基本上能同时满足实时性和精度的要求，因而也是我们进行视频压缩编码主要运用的运动估计算法。随着应用的不断扩大，对运动估计的速度和精度提出的要求也将更高，从而需要研究更好的运动估计算法以适应新的需要。

2.3.2 正交变换编码

变换编码的基本思想是通过变换操作除去由于坐标轴的选择不当而引起的相关性，而且有可能将难以处理的各种小局部相关性集中到一起处理。常见的正交变换编码方法有：KL 变换、DCT 变换、傅立叶变换、哈尔变换、Walsh-Hadamard 变换等，在图像压缩中常用 DCT 变换。

DCT 变换的进行过程是在编码端将原始图像分割成许多子像块，对每一个像块进行 DCT 变换，生成频域中的系数阵，它是一种无损压缩方法。

当以{f(X)}表示 M 个其值有限的一维实数信号序列的集合时，X = 0, 1, …, n-1，则其一维 DCT 定义为：

$$F(u)=\sqrt{\frac{2}{M}}C(u)\sum_{x=0}^{M-1}f(x)\cos\frac{(2x+1)u\pi}{2M},\ u=0,1,\ldots,M-1 \qquad (2.3)$$

一维逆变换（IDCT）定义为：

$$f(x)=\sqrt{\frac{2}{M}}\sum_{u=0}^{M-1}C(u)F(u)\cos\frac{(2x+1)u\pi}{2M},\ u=0,1,\ldots,M-1 \tag{2.4}$$

由此扩展得二维 DCT 的定义。假设数字图像 f（x, y）是具有 M 行 N 列的一个矩阵，运用 DCT 将其从空间域（xy 平面）转换到 DCT 变换域（uv 平面）：

FDCT：

$$F(u,v)=\sqrt{\frac{2}{MN}}C(u)C(v)\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}f(x,y)\cos\frac{(2x+1)u\pi}{2M}\cos\frac{(2y+1)v\pi}{2N} \tag{2.5}$$

其相应的 DCT 逆变换为：

IDCT：

$$f(x,y)=\sqrt{\frac{2}{MN}}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1}C(u)C(v)F(u,v)\cos\frac{(2x+1)u\pi}{2M}\cos\frac{(2y+1)v\pi}{2N} \tag{2.6}$$

其中 x, u=0，1，…，M-1； y, v=0，1，…，N-1；

$$C(u)=\begin{cases}1/\sqrt{2}, & u=0\\ 1, & \text{其他}\end{cases},\quad C(v)=\begin{cases}1/\sqrt{2}, & v=0\\ 1, & \text{其他}\end{cases}$$

在视频编码中，通常需要研究二维 FDCT，IDCT 的快速算法。

2.3.3 量化编码

一般而一言，量化是模拟信号到数字信号的映射，而一旦获取到视频信号并数字化后，量化则是指由数字量到数字量的多对一映射。量化器所要完成的功能是按一定的规则对表达式作近似表示，即指量化器用一组有限的实数集作为输出，其中每个数代表一群最接近于它的取样值，量化编码为有损编码。

量化常分为三类：标量量化（零记忆量化或一维量化）、向量量化（分组量化）和序列量化。在标量量化中，所有采样使用同一个量化器进行量化，每个采样的量化都和其它所有采样无关。向量量化则是从称为码本的码字集合中选出最紧密适配于序列的一个码字来近似一个采样序列（即一个向量），这种方法以输入序列与选出的码字之间失真最小为依据。

在视频编码中较多应用量化的过程是对二维 DCT 系数的量化处理，由于对于信号矩阵实施正交变换后，系数的能量分布一般比较集中，如二维 DCT 变换后的系数矩阵，能量集中在左上角，从而可想办法对于能量或能量差分重新量

化以达到信息压缩的目的。在这里，量化时对于人眼最敏感的空间频率及能量分布比较大的系数分配较多的比特数。

2.3.4 信息熵编码

信息是用不确定的量度定义的，所谓信息量则是指从 N 个相等可能事件中选出来一个事件所需要的信息度量或含量，而熵则是指将信源所有可能事件的信息量进行平均。香农信息论认为，信源所含有的平均信息量（熵）是进行无失真编码的理论极限。信源中或多或少的含有自然冗余度，这些冗余度既来自于信源本身的相关性，又来自于信源概率分布的不均匀性中，只要找到去除相关性或改变概率分布不均匀性的方法和手段，也就找到了信息熵编码的方法。这种编码也是基于统计冗余数据压缩理论的一种编码方法，为无损压缩。在多媒体视频压缩中常用的两种信息熵编码方法为：霍夫曼编码和游程编码方法。

当已被采样的视频数据拥有相同字节序列时，可以采用更紧密的序列来代替这些相同字节序列，从而实现压缩，这就是游程编码，也称为游程长编码或 RLC 编码。其主要思路是将一个相同值的连续串用一个代表值和串长来表示。例如，有一信源符合序列 66699000000，则其游程码为（6, 3），（9, 2），（0, 6）。最常见的一种情形是，当采样量化后出现大量零系数的情形，利用游程编码来表示连零码，从而降低表示零码所用的数据量。

霍夫曼编码则是这样一种编码方法，该方法能够根据已知的数据给出最佳编码，即根据已知概率决定最小位数。因此，编码字符的位长是变化的。最短的编码赋给那些最频繁出现的字符，而概率小的字符则分配较长的码字，从而提高编码效率。霍夫曼编码是一种不等长最佳编码方法，这里的最佳指对相同概率分布的信源来说，它的平均码长比其他任何一种有效编码方法都短。

在该编码方法中，先必须进行概率统计，在此基础上对所有信源中的符号赋予特定变长码字，从而求得霍夫曼表，在霍夫曼表的基础上进行变长码的编码和解码。

在视频压缩中，游程编码和 DCT 变换及霍夫曼编码方法一起使用。对分块做完 DCT 变换及量化后的频域图像作“Z”形扫描，然后进行游程编码，对其结果再做霍夫曼编码。

2.4 视频编码标准简介

近十年来，图像编码技术得到了迅速发展和广泛应用，并且日趋成熟，其标志就是几个关于图像编码的国际标准的制定，即国际标准化组织 ISO 和国际电工委员会 IEC 关于静止图像的编码标准 JPEG、国际电信联盟 ITU-T 关于电视电询会议电视的视频编码标准 H.261、H.263、H.264，和 ISO/IEC 关于活动图像的编码标准 MPEG-1, MPEG-2，MPEG-4，MPEG-7 等。这些标准图像编码算法融合了各种性能优良的图像编码方法，代表了目前图像编码的发展水平。

CCITT（即后来的 ITU-T）第 15 研究组（SGXV）于 1984 年成立了“可视电话专家组”，经过了 1985 ~ 1988 三年的研究，提出了视频编解码的 H.261 建议草案，以覆盖整个 ISDN 基群信道，满足会议电视和可视电话业务日益发展的需要。H.261 要求输入图像的格式满足 CIF（Common Intermediate Format）格式或 QCIF（Quarter CIF）格式。图像帧率最高为 29.97 帧/秒。在信道速度较低时，帧率可以降至 10 帧/秒左右。而 H.263 是为了满足近年来在普通公用电话网或移动电话网上进行可视电话通信的需要，即视频压缩码率低于 64Kbps，在诸如 28.8Kbps 等速率的信道上进行可视电话通信，ITU-T 在 H.261 建议基础上进行改革，于 1995 年 7 月提出了 H.263 建议“甚低码率通信的视频编码”。当然，H.263 建议现在也可被用于大于 64Kbps 的信道，并能产生比 H.261 好的图像。H.263 一方面以 H.261 为基础，另一方面也吸取了 MPEG 等其它一些国际标准中有效、合理的部分，如半像素精度的运动估计，P、B 帧预测等，使它的性能优于 H.261。H.264 是由国际电信标准化部门 ITU-T 和制定 MPEG 的国际标准化组织 ISO、国际电工协会 IEC 共同制订的一种视频编码国际标准格式。H.264 标准产生的初衷就是制定一个新的视频编码标准，以实现视频的高压缩比、高图像质量、良好的网络适应性。H.264 同时又被称为 MPEG-4 AVC（活动图像专家组-4 的高级视频编码）或称为 MPEG-4 Part10。H.264 的高压缩比，高图像质量和国际性的特征非常适合像网络视频应用这样在带宽受限，紧缺的应用中，H.264 大大降低了对网络带宽的需要，减轻了带宽和图像质量的矛盾所带来的运营成本的压力，同时也降低了存储，硬件设备等成本，为运营商创造了更大的盈利空间。

国际标准化组织 ISO/IEC 的运动图像专家组 MPEG（Moving Picture Experts Group）一直致力于运动图像及其伴音编码标准化工作，并制定了一系列关于一般活动图像的国际标准。其中 MPEG-1 标准就是 VCD 工业标准的核心，主要应

用于连续传输速率高达大约 1.5M bit/s 的数字存储介质，例如 CD、数字音带（DAT）和硬盘等。MPEG 组织 1995 年推出的 MPEG-2 标准是在 MPEG-1 标准基础上的进一步扩展和改进，主要针对数字视频广播、高清晰度电视和数字视盘等制定的 4~9 Mb/s 运动图像及其伴音的编码标准，MPEG-2 是数字电视机顶盒与 DVD 等产品的基础。MPEG-4 标准的制定始于 1991 年，最初的目标是制定低传输码率下的远程视频音频数据的编码方案，后来随着技术的发展，需要在各种不同的网络上传输多媒体信息，而且在无线通信中视频和音频信息越来越重要，同时消费电子（电视）、通信和计算机逐渐出现融合的趋势，Web 的兴起完全表明了交互性的重要地位。MPEG-4 的研究方向由单纯的提高压缩效率转向制订基于内容的通用的多媒体编码标准，其最重要特征是基于对象（Object-based）的编码。所谓的对象是在一个场景中能够访问和操纵的实体，具体到一幅图像中，对象就是能表征有含义的实体的一组区域(region)。对象的划分可以根据其独特的纹理、运动、形状、模型和高层语义为依据，语音、图像、视频等都可以作为单独的对象，也可以集成为更高级的对象一场景(scene)。MPEG-7 标准被称为“多媒体内容描述接口”，为各类多媒体信息提供一种标准化的描述，这种描述将与内容本身有关，允许决速和有效的查询用户感兴趣的资料。它将扩展现有内容识别专用解决方案的有限的能力，特别是它还包括了更多的数据类型。换而言之，MPEG-7 规定一个用于描述各种不同类型多媒体信息的描述符的标准集合。该标准于 1998 年 10 月提出，于 2001 年最终完成并公布。

第 3 章 H.264 的编码标准

ITU-T 和 ISO/IEC JTC1 是目前国际上制定视频编码标准的正式组织。ITU-T 的标准称之为建议，并命名为 H.26x 系列，比如 H.261、H.263 等。ISO/IEC 的标准称为 MPEG-x，比如 MPEG-1、MPEG-2、MPEG-4 等。H.26x 系列标准主要用于实时视频通信，比如视频会议、可视电话等；MPEG 系列标准主要用于视频存储(DVD) 、视频广播和视频流媒体（如基于 Internet、 DSL 的视频，无线视频等等）。除了联合开发 H.262/MPEG-2 标准外，大多数情况下，这两个组织独立制定相关标准。自 1997 年，ITU-T VCEG 与 ISO/IEC MPEG 再次合作，成立了 Joint Video Team (JVT)，致力于开发新一代的视频编码标准 H.264。1998 年 1 月，开始草案征集；1999 年 9 月，完成了第一个草案；2001 年 5 月，制定了其测试模式 TML-8；2002 年 6 月，JVT 第 5 次会议通过了 H.264 的 FCD 板；2002 年 12 月，ITU-T 在日本的会议上正式通过了 H.264 标准，并于 2003 年 5 月正式公布了该标准。国际电信联盟将该系统命名为 H.264/AVC，国际标准化组织和国际电工委员会将其称为 14496-10/MPEG-4 AVC。

3.1 H.264 标准概述

3.1.1 H.264 标准的技术背景

H.264 标准的主要目标是：与其它现有的视频编码标准相比，在相同的带宽下提供更加优秀的图像质量。

然而，H.264 与以前的国际标准如 H.263 和 MPEG-4 相比，最大的优势体现在以下四个方面：

（1）将每个视频帧分离成由像素组成的块，因此视频帧的编码处理的过程可以达到块的级别。

（2）采用空间冗余的方法，对视频帧的一些原始块进行空间预测、转换、优化和熵编码（可变长编码）。

（3）对连续帧的不同块采用临时存放的方法，这样，只需对连续帧中有改

变的部分进行编码。该算法采用运动预测和运动补偿来完成。对某些特定的块，在一个或多个已经进行了编码的帧执行搜索来决定块的运动向量，并由此在后面的编码和解码中预测主块。

（4）采用剩余空间冗余技术，对视频帧里的残留块进行编码。例如：对于源块和相应预测块的不同，再次采用转换、优化和熵编码。

3.1.2 H.264 的特征和优势

H.264 是国际标准化组织（ISO）和国际电信联盟（ITU）共同提出的继 MPEG-4 之后的新一代数字视频压缩格式，它即保留了以往压缩技术的优点和精华又具有其它压缩技术无法比拟的许多优点。

（1）低码流（Low Bit Rate）：和 MPEG-2 和 MPEG-4 ASP 等压缩技术相比，在同等图像质量下，采用 H.264 技术压缩后的数据量只有 MPEG-2 的 1/8，MPEG-4 的 1/3。显然，H.264 压缩技术的采用将大大节省用户的下载时间和数据流量收费。

（2）高质量的图像：H.264 能提供连续、流畅的高质量图像（DVD 质量）。

（3）容错能力强：H.264 提供了解决在不稳定网络环境下容易发生的丢包等错误的必要工具。

（4）网络适应性强：H.264 提供了网络适应层（Network Adaptation Layer），使得 H.264 的文件能容易地在不同网络上传输（例如：互联网，CDMA，GPRS，WCDMA，CDMA2000 等）。

3.2 H.264 编码框架

和早期的标准（例如：MPEG-1、MPEG-2、MPEG-4）一样，H.264 标准并没有明确地定义一个编码器，而是定义了编码后的比特流格式和解码的方式。实际上，不论如何，一个合适的编码器和解码器可能都包含了图 3.1 和图 3.2 所示的各个功能模块。同时，这些功能模块对于一个合适的编码器来说也是必需的，这样同时也给编码器留有很大的变化余地。这些基本的功能模块（预测，变换，量化，熵编码）与原来的标准（MPEG-1、MPEG-2、MPEG-4、H.261、H.263）在细节上有很大的不同。

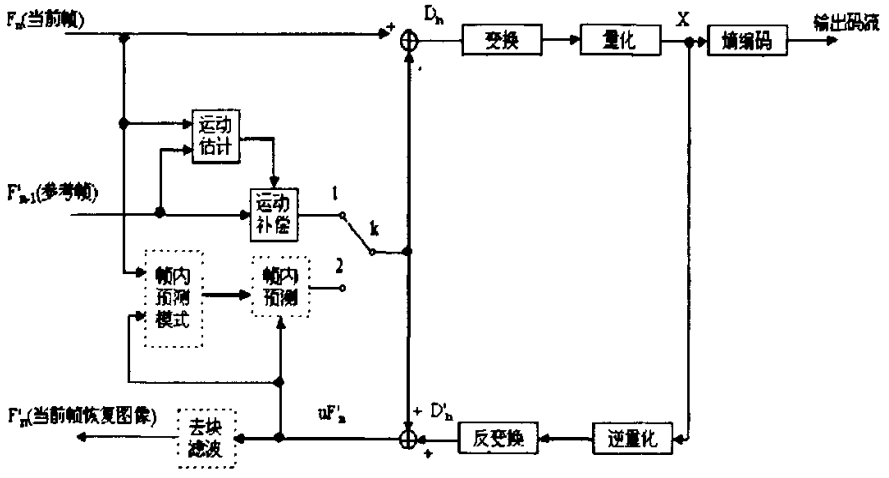


图 3.1 编码器

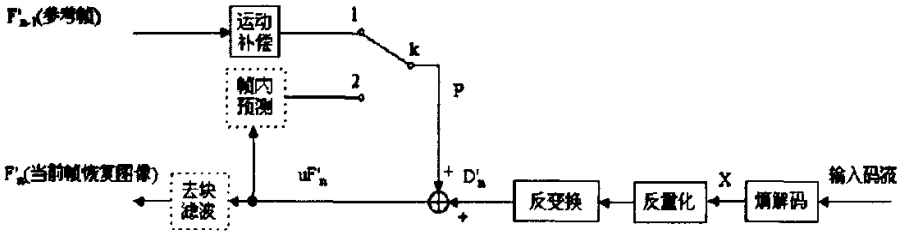


图 3.2 解码器

3.2.1 编/解码过程

由图 3.1 可以看出，H.264 的编码过程主要分为以下部分：

（1）将图像分成子图像块（宏块），以宏块作为编码单元。

（2）当采用帧内模式编码时，对宏块进行变换，量化和熵编码（或变长编码），消除图像的空间冗余。帧内模式中还增加了帧内预测模式。

（3）当采用帧间模式编码时，对帧间图像采用运动估计和补偿方法，只对图像序列中的变化部分编码，从而去除时间冗余。

解码过程为编码过程的逆过程。下面就图 3.1 编码器和图 3.2 解码器的工作流程进行详细说明：

3.2.1.1 编码器

在图 3.1 编码器框图中，Fn 代表待编码的帧，这一帧图像被分成多个 16×16 像素的宏块进行处理，每个宏块按帧内或帧间的模式进行编码，不论在哪种模式下都有一个基于参考帧重构出来的预测宏块 P。在帧内编码模式下，P 由当前帧中前面已经经过编码、解码重构模块但是没有进行滤波的宏块 uF'n 预测得到，在帧间编码模式下，P 由一个或多个参考帧进行运动补偿预测得到。在图中，参考帧用 $F'n_{-1}$ 表示，实际上参考帧可以是过去的第一帧或第二帧等或将来的第一帧或第二帧等（在时间顺序上）已经编码重构的图像。

从当前编码的宏块中减去 P 得到一个残差块 Dn，这个残差块将进行变换、量化得到 X（量化后的变换系数）这些系数将被重新排序并进行熵编码，熵编码的系数和其他的解码需要的边信息（例如：运动预测的模式量化器的步长和描述宏块如何进行运动补偿的运动向量信息等）一起形成比特流，比特流经过 NAL（Network Abstraction Layer）层进行传输或存储。

在编码器中量化后的系数 X 将被解码重构，以便为对将来的宏块进行编码时使用。系数 X 将通过逆量化和逆变换产生一个差分宏块 D'n。

差分宏块 D'n 和原始的宏块之间并不是完全一样的，因为经过量化运算后，会产生量化误差。预测宏块 P 和 D'n 进行加法运算得到一个重构宏块 uF'n，通过一个滤波器以减少块失真得到一个重构图像 F'n。

3.2.1.2 解码器

解码器从 NAL 层中接收到压缩后的比特流。数据元素进行熵解码，然后重新排序，恢复出来量化后的系数 X， X 再经过逆量化和逆变换得到 D'n 通过从比特流中的解码出来的头信息，解码器产生一个预测块 P，P 的产生过程和编码过程一样。P 和 D'n 相加得到一个 uF'n 最后再经过滤波器得到恢复图像 F'n。

在编码其中的重构路径是为了和在解码器中的重构路径一样，以便产生相同的预测块 P。如果不是这样的话，将会导致在编码器和解码器中的 P 不同，从而导致附加的错误和漂移。

3.2.2 编码器的组成部分

下面将就编码的各个组成部分作详细说明：

3.2.2.1 图像格式定义

H.264 国际标准支持的图像格式有：

公用中分辨率格式 CIF（Common Intermediate Format）（352×288 像素）和 1/4 公用众分辨率格式 QCIF（Quarter Common Intermediate Format）（176×144 像素）。

表 3.1 图像文件格式

	CIF 格式		QCIF 格式	
	行数/帧	像素/行	行数/帧	像素/行
亮度（Y）	288	352	144	176
色度（Cr）	144	176	72	88
色度（Cb）	144	176	72	88

3.2.2.2 色度信号采样

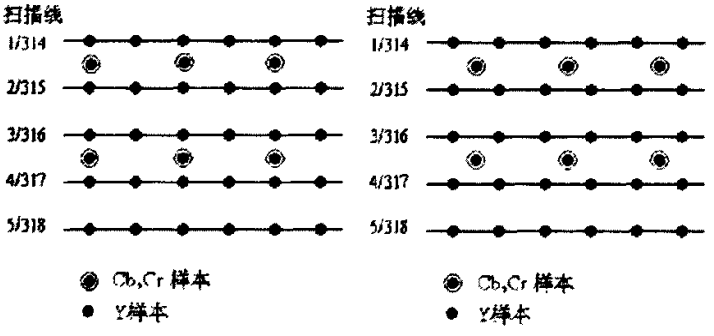


图 3.3 采样格式

(a) H.264 采用的采样格式　　(b) H.263 采用的采样格式

在采样过程中，Y 样本的位置就是像素点所在的位置，在水平方向和垂直方向上相邻的 4 个像素点的中间位置采样一个 Cb 和 Cr 样本。虽然 H.264 和 H.263 使用的也是 4：2：0 的采样格式，但是它们的含义与其他标准有所不同。与 H.263 相比，H.264 的子采样在水平方向上没有半个像素的偏移，见图 3.3b。

3.2.2.3 图像分块和编码顺序

采样后的视频图像都被分成 16×16 的宏块，每个宏块包括 1 个亮度子块和 2 个 8×8 的色度子块。一帧 CIF（352×288）格式的图像分为 18×22=396 个宏

块，一帧 QCIF（176×144）格式的图像被分为 9×11=99 个宏块。若干个宏块又组成了宏块组，称为片（Slice）。一幅 QCIF 的图像最少由 1 个片组成，最多由 99 个片组成。宏块的划分方式如图 3.4 所示：

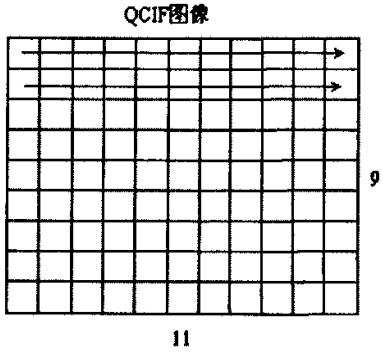


图 3.4 宏块的划分方式

3.2.2.4 运动估计和运动补偿

H.264 的特征是：加大了预测部分的比重，通过改善预测误差而提高编码效率。

① 运动补偿块大小采用可变形式，可从 16×16，16×8， 8×16， 8×8，8×4，4×8，4×4 中选择采用不同的块大小的运动矢量预测可以比单独 16×16 块的预测方法提高大于 15%的编码率；

② 运动矢量的精度目前可达 1/4 或 1/8 像素，与整数精度的空间预测相比，可以提高大于 20%的编码率；

③ 采用多参考帧进行帧间预测，这样比单独参考帧方法可以节省 5%～10%的传输码率，并且有利于码流的错误恢复。

3.2.2.5 变化编码

H.264 残差图像的亮度系数采用 4×4 像素大小的块进行变换编码，与传统的 8×8 大小的浮点型 DCT 变换不同，它使用一种整型变换编码。这种方法避免了取整误差和反变换误匹配的问题。另外，H.264 对系数扫描的方式有两种，除了和 H.263 相同的简单扫描方式外，还增加了双扫描方式。

3.2.2.6 熵编码

H.264 采用两种方法进行熵编码：CAVLC 编码和 CABAC 编码算法。采用

基于上下文的自适应二进制算术编码算法（CABAC）能够充分利用上下文信息和算术编码的优点，使得编码后的平均码长更逼近图像的信息熵，达到最佳的编码效率。采用 CABAC 算法进行编码可以提高大约 10%的编码率。

3.2.2.7 码率控制

由于采用了熵编码，产生的比特流的速率是随着视频图像的统计特性不同而变化的。但大多数情况下传输系统分配的频带都是恒定的，因此在编码比特流进入信道前需设置信道缓存。码率控制的目的就是根据缓存器的状态和信道速率来控制编码器，使得编码输出的码流尽可能稳定，这样编码端缓存器也不易出现上溢现象，从而有效地减少和避免跳帧现象，获得更好的重建图像质量。

传统视频编码码率控制策略是依据缓存器的状态和信道速率来调整量化步长，当编码器的瞬时输出速率过高，缓存器将要上溢时，就使量化步长增大以降低编码数据速率，当然也相应增大了图像的损失；当编码器的瞬时输出速率过低，缓存器将要下溢出时，就使量化步长减小以提高编码数据速率。典型的 H.261 编解码器采用对传输缓冲器占有率与反馈的量化系数之间以线性关系描述，并据此来控制传输缓冲器输入码流的速率。其主要缺陷在于采用了线性的方法控制一个非线性动态的视频编码器传输缓冲器，在图像数据突变时，由于调节过慢有可能使得图像质量下降甚至影响系统的正常运行。

3.2.3 H.264 的框架

在 H.264 中定义了 3 个框架，每个框架支持一系列的特定的编码功能。编码器和解码器都必须遵守这些规定。

基线框架（Baseline Profile）支持帧间和帧内编码，支持 I 帧和 P 帧，支持 CAVLC 等。它的主要应用是可视电话、视频会议、无线通信等。

主框架（Main Profile）包括支持交错视频，支持 B 帧，帧间编码时使用权重预测，熵编码使用 CABAC。它的主要应用是视频存储和电视广播。

扩展框架（Extended Profile）不支持交错视频和 CABAC，但增加了一些在进行比特流切换时有效的帧模式 SI（Switching I）帧和 SP（Switching P）帧，能够有效的提高从错误中恢复的能力。它的主要应用是流媒体应用。

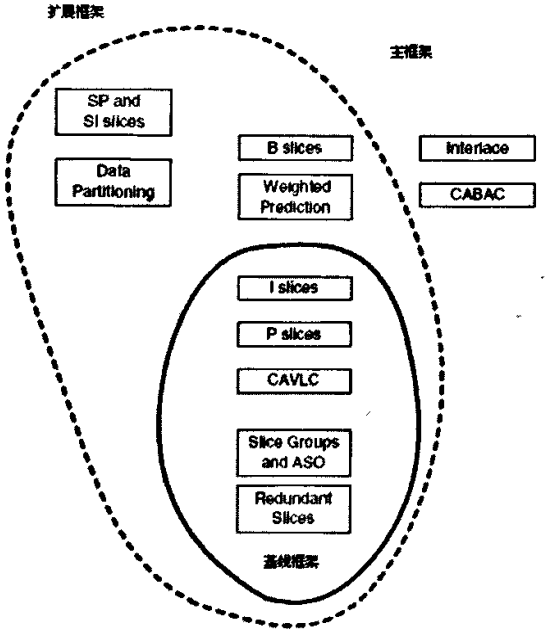


图 3.5 H.264 中的基线框架、主模式和扩展框架

表 3.2 H.264 的帧编码模式

帧类型	描述	支持的框架
I（Intra）	只包含帧内预测的宏块（I）	全部
P（Predicted）	包含帧间预测宏块（P）和 I 型宏块	全部
B（Bi-Predictive）	包含帧间双向预测宏块（B）和 I 型宏块	扩展和主要
SP（Switching P）	利于在编码的比特流中切换，包括 I 和 P 宏块	扩展
SI（Switch I）	利于在编码的比特流中切换，包含 SI 宏块（一种特殊的帧内编码宏块）	扩展

3.3 H.264 的视频流语法

在 H.264 的标准文件中详细规定了视频流的语法结构。 因为本文重点是对 H.264 的编码算法进行应用，所以在此只是简单介绍一下 H.264 的视频流语。

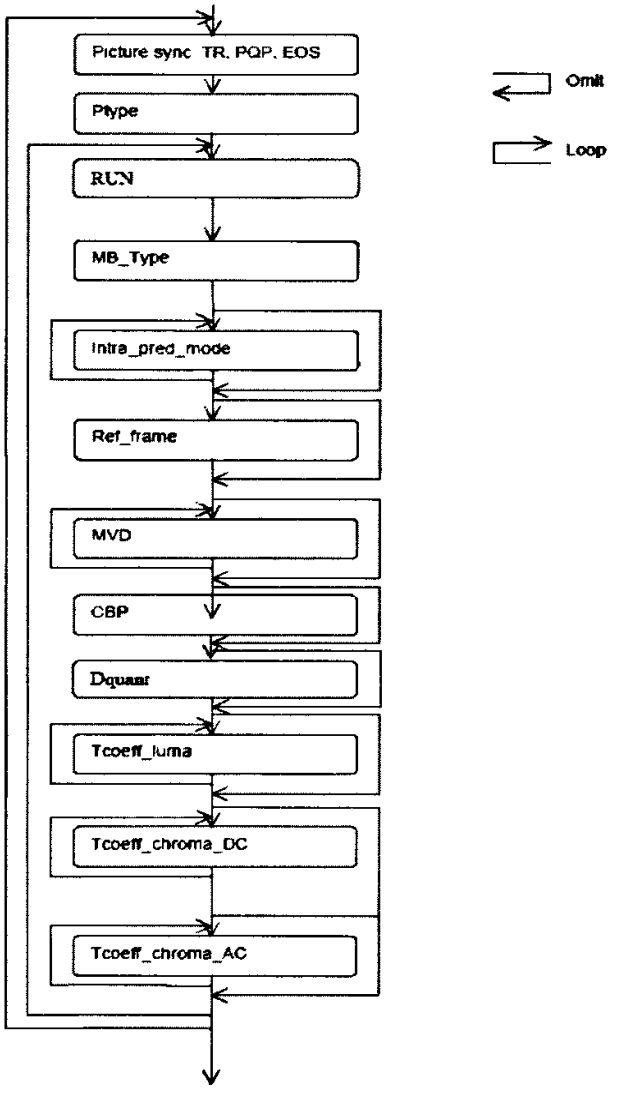


图 3.6 H.264 的视频流语法框图

语法元素的具体含义：

Picture Sync：图像头和同步信息。31 比特长。如图 3.7 所示

8b TR	5b PQP	1b Formats	1b: EOS

图 3.7 Picture Sync 元素结构图

TR（8bits）：时间参考。由前面的时间参考帧的头信息加上其后传送过的

非参考帧或跳过的帧数决定。

PQP（5bits）：量化步长。存储关于量化器的亮度系数量化步长信息，这 5 个比特是自然二进制数，代表量化步长，范围是 0-31。

Formats（1bit）：图像格式。0 代表 QCIF, 1 代表 CIF。

EOS（1bit）：结束标志。0 代表图像头，1 代表序列结束。

Ptype：图像编码模式，它的值和图像的编码模式关系如下：

① 0：只参考最近恢复图像的帧间预测模式；

② 1：多参考帧的帧间预测模式，在这种情况下，必须为每个宏块标明用来预测的参考帧信息；

③ 2：帧内编码模式；

④ 3：值参考最近恢复图像的 B 帧编码模式；

⑤ 4：多参考帧的 B 帧编码模式。对于这种模式，也必须为每个宏块标明用来预测的参考帧信息。

RUN：如果一个宏块没有信息需要传送则这个宏块被标注 Skipped，如果该宏块被标注 Skipped，则将前一帧解码的图像相应宏块复制到当前帧；Run 表示 P 或 B 图像中跳过的宏块个数。

MB_Type：（宏块类型，Macro block type）对于帧内模式，宏块类型分为 Intra 4×4 和 Intra 16×16；两种对于 P/SP 帧和 B 帧模式，宏块类型分为 skip、NXM 和 Intra 4×4 和 Intra 16×16 等。

Intra_pred_mode：帧内预测模式。

Ref_frame：用来标注参考帧的位置：

① 0：最近解码的一帧图像 (1 Frame Back)；

② 1：当前帧的前面第 2 帧图像 (2 Frame Back)；

③ 2：当前帧的前面第 3 帧图像 (3 Frame Back)；

等等。

MVD：运动矢量(Motion Vector Data)，根据宏块的编码模式，传输相应的运动矢量数据。

CBP：编码块模式(Coded Block Pattern)，表示哪个 8×8 大小的亮度和色度块包含变换系数的参数。一个 8×8 的块包括 4 个 4×4 的块，当一个 8×8 的块有系数需要传输也就是说至少一个 4×4 的块有系数需要传输。在 CBP 中有 4 个比特用来表示哪一个块包含非零系数这 4 个比特叫做 CBPY。如果一个 8×8

块中包含至少一个非零系数，则相应的字节位置 1，否则该位为 0，表示该 8×8 块内没有系数传输。

色度块编码模式定义如下：

① nc=0：没有色度系数；

② nc=l：存在非零的 2×2 色度块直流变换系数，所有的色度块交流系数为 0，因此也不传输 AC 系数的 EOB 标志位；

③ nc=2：可能存在非零的 2×2 色度块直流变换系数，同时存在至少一个非零的色度块交流系数。此时，需要传输 10 个 EOB 标志位（2 个 DC 系数的标志位、2×4=8 个 4×4 块的色度交流系数的标志位）；

一个宏块总计的 CBP=CBPY + 16Xnc。

Dquant：宏块级的量化步长。当没有非零变换系数传输时，Dquant 不存在。当 CBP 表明宏块中有非零系数或对宏块采用 16×16 的帧内编码模式时，Dquant 才存在。Dquant 的范围是（16×16）。修改后的量化步长，记为 $QUANT_{new}$，$QUANT_{new}=modulo_{32}(QUANT_{old}+Dquant+32)$。

Tcoeff_luma：亮度传输系数。

Tcoeff_chroma_DC：直流色度传输系数。

Tcoeff_chroma_AC：交流色度传输系数。

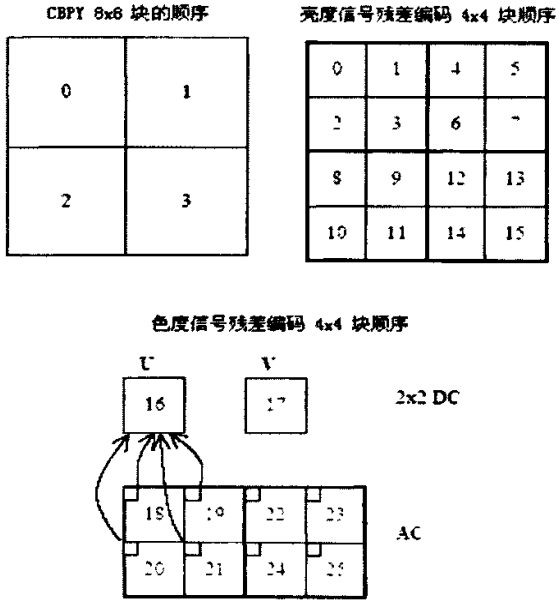


图 3.8 CBPY 块排列顺序和 4×4 残差编码块排列顺序

第 4 章 基于有线网络的图像传输技术的应用

前几章我们对图像传输系统的体系结构及一些关键技术作了介绍，下面将具体设计和实现远程视频监控系统。该系统采用的是典型的套接字客户机/服务器结构，它由服务器端的监控现场的视频数据处理模块、视频数据发送模块和客户端的视频数据接收播放模块 3 个部分组成。服务器端和客户端的运行界面分别如图 4.1、图 4.2 所示。

图 4.1 服务器端界面

图 4.2 客户端界面

视频数据处理模块运行在监控现场的服务器端主机上，主要实现视频数据的实时采集和动态存储、视频图像的实时播放以及视频文件的播放控制。

视频数据发送模块运行在监控现场的服务器端主机上，负责对视频压缩卡的网络发送进行管理，并在计算机网络中组播发送视频流，使网络上的其它计算机可以通过软件接收视频信号。

视频数据接收播放模块作为视频数据发送模块的接收端，独立运行在客户端主机上，它接收视频流加以保存并实时播放显示。

视频数据发送和接收的具体过程如下：服务器端先启动，并一直处于监听状态，当客户端请求数据时，它首先向相应的服务器端发出数据请求，服务器端在接收到请求后将它的 IP 组播地址和端口号传给客户端，客户端收到后，再向服务器端发回确认信息，同时加入该组播组，等待接收数据。服务器端收到确认信息后向该组播发送视频数据。视频数据流在该系统中的流程如图 4.3 所示。

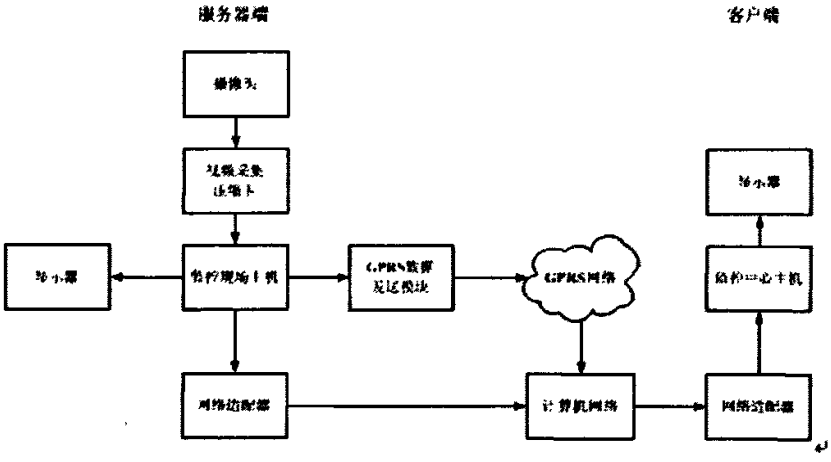


图 4.3 视频数据流流程图

4.1 视频采集与压缩

在监控现场由摄像机得到的是模拟的视频信号，要将这些模拟的视频信号交由计算机处理并在数字通信线路上传输，必须采用相应的设备将模拟信号转化为数字信号并且进行压缩。DS-4004HC 是海康威视公司开发的一款高性能的视频采集压缩卡，通过它可以将模拟视频信号、音频信号实时数字化并压缩编码，然后交由计算机直接处理。而且利用 DS-4004HC 提供的 SDK 可以很方便的进行二次开发。

4.1.2 DS-4004HC 视频采集卡简介

DS-4004HC 是一款采用 DSP 专用芯片和先进结构算法、具有出色实时性能的 H.264 视音频采集压缩卡。它所输出的顶级质量的视音频数据特别适用于网络视频和多媒体应用。DSP 处理芯片内嵌入了高性能的 H.264 图像压缩处理模块和运动补偿处理模块作为核心部件，视频处理处理能力十分强大，实施对快速运动的画面也能达到理想的压缩效果。

DS-4004HC 提供每秒 25/30 帧实时高质量的视频/音频同步采集/压缩，不但能够适用于 Video CD 和多媒体视频制作的需要而且能完全满足 Internet/Intranet 上的视频通讯要求和低带宽的无线视频通讯要求，视频采集数据从 32K bit/s 到

4M bit/s 覆盖了从 Video CD 制作到网络视频通讯全线应用。

对于视频多媒体应用开发的用户，DS-4004HC 提供了完整的开发工具包。可以把 DS-4004HC 的操作嵌入用户的系统软件中。在开发视频网络应用时，只需要很少的代码就可以实现对 DS-4004HC 的控制，而且，DS-4004HC SDK 的代码结构可以实现在一台机器上插多块 DS-4004HC，用不同的进程分别操作，从而实现视频服务器端的开发。

DS-4004HC 的主要技术参数如表 4.1 所示。

表 4.1 DS-4004HC 的主要技术参数

参数	说明
视频压缩标准	H.264。.压缩比极好，图像质量好。
视频输入	4 路复合视频信号
支持格式	PAL、NTSC
分辨率	PAL：　176×144(QCIF)、352×288(CIF)、704×288(2CIF)、528×384（DCIF）、704×576（4CIF） NTSC：176×120(QCIF)、352×240(CIF)、704×240(2CIF)、528×320（DCIF）、704×480（4CIF）
帧率	25F/S（PAL）、30F/S（NTSC）
输出码率	32Kbps-1000K bps(CIF)；70K bps-4000K bps(4CIF)
音频压缩标准	OggVorbis
语音输入	4 路语音线路输入
采样率	16K Hz
输出码率	16K bps
应用	一块 DS-4004HC 压缩卡上可实现 4 路 CIF 或者 2 路 4CIF 分辨率的实时编码，也支持 4 路 4CIF 的非实时编码；兼容的 SDK 接口使开发更加快捷、容易；一台 PC 能稳定、可靠地支持 64 路视音频输入，每路都能进行实时编码，控制参数完全独立可调。

4.1.2 DS-4004HC 二次开发基础

DS-4004HC 提供的二次开发包是动态链接库形式的，利用此开发包可以完成的功能包括采集 H.264 数据流到文件或者应用程序的缓冲区，并且通过 DirectX 等可支持视频流的实时播放。

4.1.2.1 设备的初始化和关闭

设备的初始化首先调用 InitDSPs()初始化板卡，应在应用软件启动时完成。然后调用 GetTotalChannels()获得系统内可适用的通道个数。再调用 ChannelOpen()打开通道，获取相关的操作句柄，在后续的与通道相关的操作必须使用该句柄。

当不再使用设备时，调用 ChannelClose()关闭通道，释放相关资源，以便其它应用程序可以使用，然后调用 DeInitDSPs()关闭板卡上功能，退出应用程序。

4.1.2.2 使用设备

设备初始化建立后，将从设备中得到视频流数据，对得到的视频流数据可以直接写到文件里，或由回调函数来处理。回调函数是由系统自动调用的函数。需要按照指定的格式编写自己的回调函数，它的主要作用是在满足某种特定的条件时，执行相应的操作。如在采集卡刚开始采集数据时，打开要写的文件；当采集到一定数量的数据时，对采集的数据流进行各种处理，如保存到指定文件里或者写到程序设置的缓冲区中。

使用视频卡的过程如下：在初始化板卡并且打开视频通道以后，通过 API 函数 SetPreviewOverlayMode()设置视频预览模式，随即可启用 StartVideoPreview()函数对监控视频进行本地预览，然后可以通过一系列函数对视频信号的属性设置和更改，调用 StartVideoCapture()和 StopVideoCapture()启动和停止主通道数据截取（或者调用 StartsubVideoCapture()和 StopsubVideoCapture()启动和停止子通道数据截取）。最后可调用 StopVideoPreview()来停止视频图像预览。

4.1.2.3 DS-4004HC 二次开发流程图

DS-4004HC 二次开发流程图如图 4.4 所示。

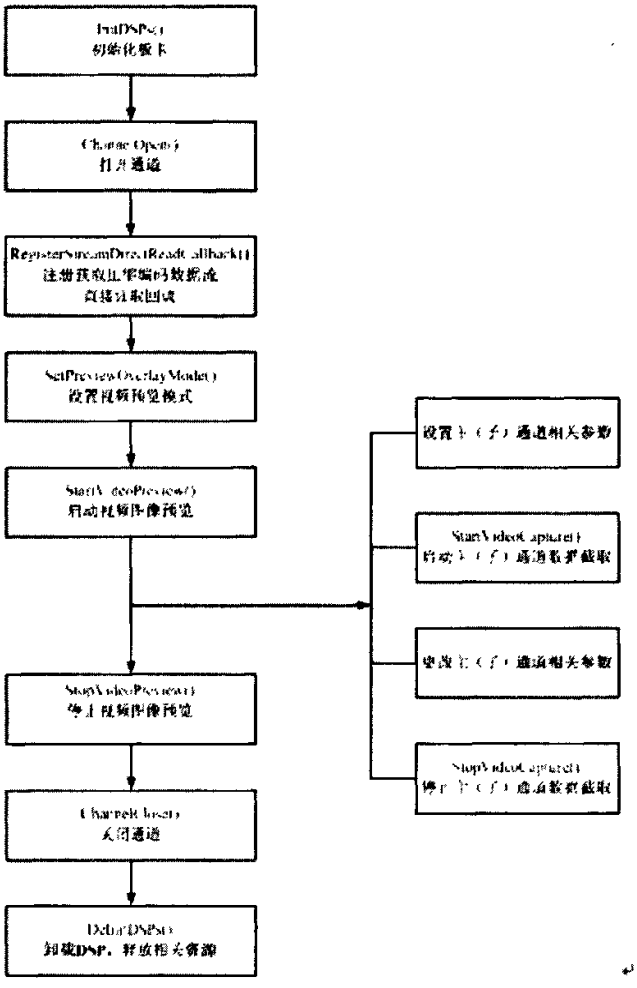


图 4.4 DS-4004HC 二次开发流程图

4.1.3 视频采集的相关程序说明及实现

视频采集模块的菜单包括采集参数设置、开始主通道录像、停止主通道录像、开始子通道录像、停止子通道录像、设置、抓图，如图 4.5 所示，相对应的消息处理函数分别为 OnSetting()、OnStart()、OnStop()、OnStartSub()、OnStopSub()、OnSetting()、OnCapimg()。另外，定义了 InitDriver()、ClearDriver()、StreamDirectReadCallback()等函数对实现对 DS-4004HC 视频采集卡的操作。下面将对各个函数作详细说明，这些函数都定义在 SenderDlg.h 文件中。

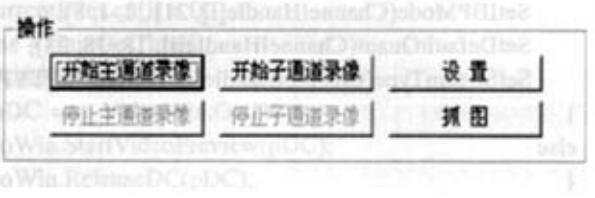


图 4.5 视频采集模块功能键

4.1.3.1 InitDriver()函数和 ClearDriver()函数

InitDriver()函数实现了用 DS-4004HC 进行数据采集时的一些初始化工作，包括设备驱动的打开，各项参数的设定，工作模式的选择，注册数据流处理回调函数，以及打开本地视频播放。ClearDriver()函数实现了结束数据采集时所做的一些工作，可理解为 InitDriver()函数的逆操作。包括停止采集数据、关闭数据流，释放设备和相关资源等。

```
BOOL CSenderDlg::InitDriver()
{
    //初始化板卡上 DSP;
    if(InitDSPs() < 0)
    {
        AfxMessageBox("Can not initialize DSPs !\n");
        return FALSE;
    }
    CString dsps;
    dsps.Format(" %d Channels are Found!", GetTotalDSPs());
    //将视频窗口与 m_VideoWin 对象相关联;
    m_VideoWin.SubclassDlgItem(IDC_VIDEOWIN, this);
    //打开视频通道;
    for(i = 0; i < GetTotalDSPs(); i++)
    {
        ChannelHandle[i] = ChannelOpen(i);
        if (ChannelHandle[i]<0)
            AfxMessageBox("Channel open error > 0");
        else if (ChannelHandle[i] ==(HANDLE) 0xffff)
            AfxMessageBox("Channel open error 0xffff");
        gChannelTotalLength[i] = 0;
    //根据不同的网络连接方式，设置帧类型、视频质量、视频流类型;
        if (servertype == DIALTYPE)
```

```
            {
                SetIBPMode(ChannelHandle[i], 211, 2, 1, 8);
                SetDefaultQuant(ChannelHandle[i], 18, 18, 23);
                SetStreamType(ChannelHandle[i], STREAM_TYPE_VIDEO);
            }
            else
            {
                SetIBPMode(ChannelHandle[i], 100, 2, 1, 25);
                SetDefaultQuant(ChannelHandle[i], 15, 15, 20);
            }
        }
        if (servertype == DIALTYPE)
        {
            for(i = 0; i < GetTotalDSPs(); i++)
                SetEncoderPictureFormat(ChannelHandle[i],
ENC_QCIF_FORMAT);
        }
        else
        {
            for(i = 0; i < GetTotalDSPs(); i++)
            {
                if ( i==0 )
                {
                //初始化时，设置通道 1 为 4CIF 格式，其他通道为 CIF 格式；
                 SetEncoderPictureFormat(ChannelHandle[0],
ENC_4CIF_FORMAT);
                 bEncodeCifAndQcif[0] = FALSE;
                }
                else
          SetEncoderPictureFormat(ChannelHandle[i], ENC_CIF_FORMAT);
            }
        }
        //注册数据流直接回调函数；
        RegisterStreamDirectReadCallback(::StreamDirectReadCallback,this);
        RegisterMessageNotifyHandle(m_hWnd, MsgDataReady);
        SetOverlayColorKey(gBackgroundColor);
        //设置定时器；
        gTimer = SetTimer(1, 1000, 0);
        SetTimer(2,2000,0);
        //初始化文件参数；
```

```
    for (i=0;i<MAX_CHANNELS;i++)
        gCurrentFileLen[i] = 0;
    StartTime = timeGetTime();
    //开启本地视频播放;
    CDC *pDC = m_VideoWin.GetDC();
    m_VideoWin.StartVideoPreview(pDC);
    m_VideoWin.ReleaseDC(pDC);
    return TRUE;
}
```

4.1.3.2 StreamDirectReadCallback()函数

StreamDirectReadCallback()是一个回调函数，实现对视频采集卡传递进来的数据流的处理。它先将数据压入一个视频流缓冲区队列中，如果系统现在在采集数据，则将数据同时写入到采集文件中去。根据初始设置的参数，如果采集文件满，则重新建一个文件保存视频数据。其函数声明如下：

int __cdecl StreamDirectReadCallback(ULONG channelNum, void *DataBuf, DWORD Length, int frameType, void *context)。

4.1.3.3 OnSetting()函数

OnSetting()函数用于设置视频的相关参数，设置主通道的编码分辨率格式、编码帧的结构和帧率、码流的控制模式和最大比特率、单个文件大小等等，以及是否添加 OSD 信息和相关图标于视频图像上。当单击主操作界面上的 Setting 键，将弹出参数设置窗口如下。

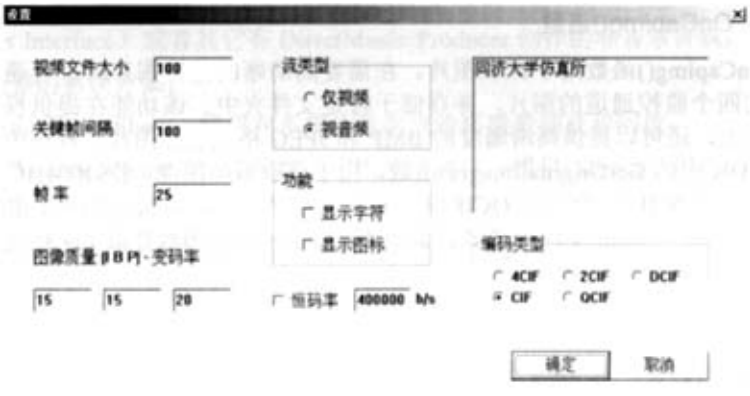


图 4.6 参数设置对话框

系统采集编码的过程中，不能调用该函数，因为新的参数值要在重新调用StartVideoCapture()函数后方能生效。函数响应时，先将系统预设定的各项参数值放入即定的变量中，然后弹出设置窗口，将各个预设值显示在窗口中，用户可针对个人需要调整各个量值，确定退出后，程序调用 StartVideoCapture()，系统将按照新的参数值重新开始视频编码。

4.1.3.4 OnStart()函数和 OnStop()函数

OnStart()函数相应用户界面的“开始主通道录像”采集文件设置命令，它先建立一个以当前时间为文件名的 H.264 格式的视频文件，然后调用StartVideoCapture()函数启动主通道数据截取（录像）。随着该函数的调用，系统开始适时调用回调函数 StreamDirectReadCallback()，它按照默认设置或者用户的后期设置保存数据采集文件，根据文件大小自动结束当前文件，同时重新建立一个文件存储视频数据。

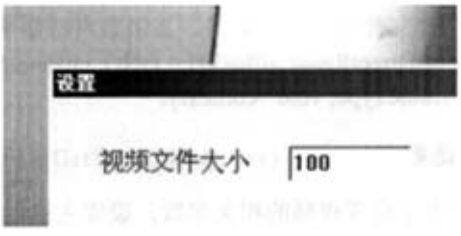


图 4.7 视频文件大小设置界面

OnStop()函数相应停止保存视频命令，关闭并保存当前视频文件。

4.1.3.7 OnCapimg()函数

OnCapimg()函数用于抓取图片。在需要高清晰的监控图片时，可通过此函数抓取四个监控通道的图片，并存储于特定文件夹中。该功能在提供视频图像的基础上，还可以提供高清晰度的 BMP 和 JPEG 格式监控图片。程序中主要用到了 SDK 中的 GetOriginalImage()函数，用于获取原始图像。DS4004HC 原始图像是标准的4CIF格式(包括QCIF编码)，用户程序可再调用SaveYUVToBmpFile)来生成 24 位的 bmp 文件；或者，调用 GetJpegImage()函数抓取 jpg 格式图像。此处省略相关实现代码。

4.2 实时图像和视频文件的播放

实时图像和视频文件的播放都是通过微软公司提供的 DirectShow 组件实现

的。DirectShow 提供对多媒体数据流的高质量捕获和回放。使用 DirectShow 开发应用程序时，既可以直接调用它所提供的 COM 接口来完成一定的功能，也可以在它的基础上自己编写 COM 接口。在这里，利用 DirectShow 提供的 COM 接口设计实现一个 CVideoWin 类，这个类封装了一系列函数来提供对实时图像和视频文件的播放操作。

4.2.1 DirectShow 简介

DirecShow 是微软推出的 DirectX 多媒体软件开发包中的一个组件。DirectX 是 Microsoft 公司为游戏和其他高性能多媒体应用所提供的一套底层应用程序编程接口。这些接口包括对二维和三维图形、声效和音乐、输入设备以及多玩家网络游戏等的支持。目前 DirectX 的最高版本是 DirectX 9.0。DirectX 9.0 由下列组建组成：

（1）DirectX Graphics：该组件组合 DirectX 旧版本中的 DirectDraw 和 Direct3D 两个组件，使其成为一个适用于所有图形程序的单独的应用程序接口；其中的 Direct3D 扩展（D3DX）应用程序库简化了多数图形程序的工作；

（2）DirectInput：支持各种输入设备，完全支持力反馈技术；

（3）DirectPlay：支持多玩家网络游戏；

（4）DirectSound：支持用于播放和捕获音频波形的高性能音频应用软件的开发；

（5）DirectMusic：为音乐音轨以及基于波表、MIDI（Musical Instrument Devices Interface）或者其它有 DirectMusic Producer 创作的非音乐音轨，提供了一套完整的解决方案；

（6）DirectShow：提供对多媒体数据流的高质量捕获和回放；

（7）DirectSetup：一个简单的应用程序接口，提供 DirectX 组件的自动安装；

（8）DirectX Media Objects：提供对数据流对象的读写支持，包括视频和音频的编解码器及其效果；

DirectShow 技术是建立在 DirectDraw 和 DirectSound 组件基础之上的，它通过 DirectDraw 对显卡进行控制以显示视频，通过 DirectSound 对声卡进行控制以播放声音。DirectShow 可提供高质量的多媒体流的捕获和回防功能；支持多种

媒体格式，包括 ASF（Advanced Systems Format），MPEG（Motion Picture Experts Group），AVI（Audio-Video Interleaved），MP3（MPEG Audio Layer-3）和 WAV 声音文件；可以从硬盘上捕获媒体数据流；可以自动检测并使用视频和音频加速硬件。因此，DirectShow 可以充分发挥媒体的性能，提高运行速度，可以简化媒体播放、媒体间的格式转换和媒体捕获等工作。同时，它还具有极大的可扩展性和灵活性，可以由用户自己创建组建，并将这个组建加入 DirectShow 结构中以支持新的格式或特殊的效果。

应用程序与 DirectShow 组件以及 DirectShow 所支持的软硬件之间的关系如图 4.8 所示。

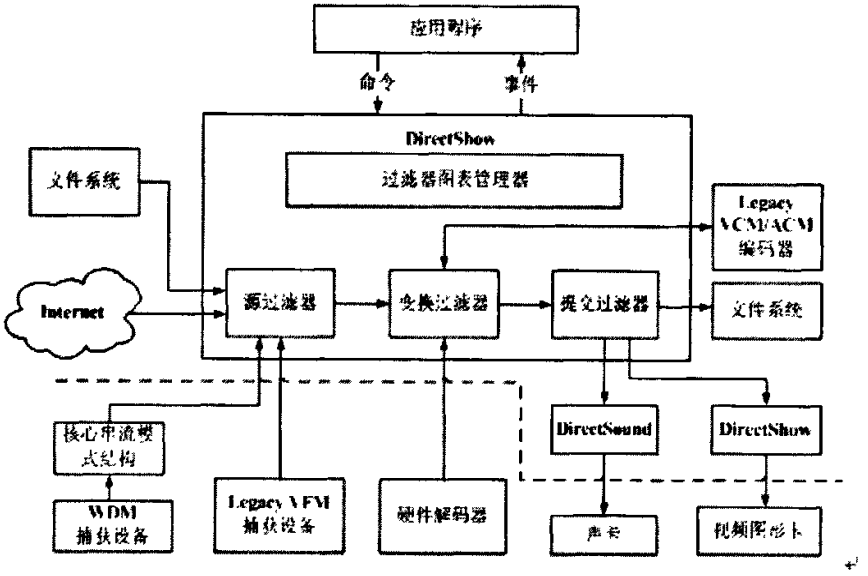


图 4.8 DirectShow 系统框图

4.2.1.1 过滤器（filter）

过滤器主要分为以下几种类型。

（1）源过滤器（source filter）：源过滤器引入数据到过滤器图表中，数据来源可以说是文件、网络、照相机等。不同的源过滤器处理不同类型的数据源；

（2）变换过滤器（transform filter）：变换过滤器的工作是获取输入流，处理数据，并生成输出流。变换过滤器对数据的处理包括编码器、格式转换、压缩解压缩等；

（3）提交过滤器（render filter）：提交过滤器在过滤器图表里处于最后一级，

它们接受数据并把数据提交给外设；

（4）分割过滤器（splitter filter）：分隔过滤器把输入流分割成多个输出，例如，AVI 分割过滤器把一个 AVI 格式的字节流分割成视频和音频流；

（5）混合过滤器（mix filter）：混合过滤器把多个输入组合成一个单独的数据流，例如，AVI 混合过滤器把视频流和音频流合成一个 AVI 格式的字节流；

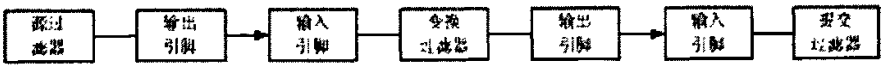
过滤器的这些分类并不是绝对的，例如一个 ASF 读过滤器（ASF Reader filter）既是一个源过滤器又是一个分割过滤器。

在 DirectShow 里，一组过滤器称为一个过滤器图表（filter graph）。过滤器图表用来连接过滤器以控制媒体流，它也可以将数据流返回给应用程序，并搜索所支持的过滤器。过滤器有 3 种可能的状态：运行、停止和暂停。

4.2.1.2 引脚（pin）

过滤器可以和一个或多个过滤器相连，连接的接口也是 COM 形式，称之为引脚。过滤器利用引脚在各个过滤器间传输数据。每个引脚都是从 Ipin 这个 COM 对象派生出来的，且都是过滤器的私有对象，过滤器可以动态地创建引脚、销毁引脚和自由控制引脚的生存时间。引脚可以分为输入引脚（Input pin）和输出引脚（Output pin）两种类型，两个相连的引脚必须是不同种类，即输入引脚只能和输出引脚相连，且连接的方向总是从输出引脚指向输入引脚。

过滤器之间的连接（也就是引脚之间的连接），实际上是连接双方媒体类型（Media Type）协商的过程。连接的大致过程为：如果调用连接函数是已经指定了完整的媒体类型，则用这个媒体类型进行连接，成功与否都结束连接过程；如果没有指定或不完全指定了媒体类型，则进入下面的枚举过程——枚举欲连接的输入引脚上所有的媒体类型，逐一用这些媒体类型与输出引脚进行连接（如果连接函数提供了不完全媒体类型，则要先将每个枚举出来的媒体类型与它进行匹配检查），如果输出引脚也接受这种媒体类型，则引脚之间的连接宣告成功；如果所有输入引脚枚举的媒体类型，输出引脚都不支持，则枚举输出引脚上的所有媒体类型，并逐一用这些媒体类型与输入引脚进行连接，如果输入引脚接受其中的一种媒体类型，则引脚之间的连接宣告成功；如果输出引脚上的所有媒体类型，输入引脚都不支持，则这两个引脚之间的连接过程宣告失败。过滤器与引脚连接如图 4.9 所示。



4.9 过滤器和引脚连接示意图

4.2.1.3 媒体类型（Media Type）

媒体类型是描述数字媒体格式的一种通用的可扩展方式。两个过滤器相连时，必须使用一致的媒体类型，否则这两个过滤器就不能相连。媒体类型能识别上一级过滤器传送给下一级过滤器的数据类型，并对数据进行分类。

实际在很多应用程序中，用户根本不需要担心媒体类型的问题，DirectShow 会处理好所有的细节。但有些应用程序需要对媒体类型进行操作。媒体类型一般可以由两个表示：AM_MEDIA_TYPE 和 CMediaType。前者是一个结构，后者是从这个结构继承过来的类。

每个 AM_MEDIA_TYPE 由 3 个部分组成：Major type、Subtype 和 Format type。这 3 个部分都使用 GUID（全局唯一标示符）来唯一标识。主要定性描述一种媒体类型，这种媒体类型可以使视频、音频、比特数据流或 MIDI 数据等；Subtype 进一步细化了媒体类型，如果是视频的话可以进一步指定是 RGB-24，还是 RGB-32 或是 UYVY 等等；Format type 则用一个结构更进一步细化媒体类型。

如果媒体类型的 3 个部分都制定了某个具体的 GUID 值，则称这个媒体类型是完全指定的；如果媒体类型的 3 个部分中有任何一个值是 GUID_NULL，则称这个媒体类型是不完全指定的。GUID_NULL 具有通配符的作用。

5.2.1.4 过滤器图表管理器（Filter Graph Manager）

DirectShow 通过过滤器图表管理器来控制过滤器图表中的过滤器。过滤器图表管理器是 COM 形式的，它的功能有协调过滤器间的状态转变、建立参考时钟、把事件（event）传送给应用程序和为应用程序提供建立过滤器图表的方法。

一些常用的过滤器图表管理器接口如下：

IGraphBuilder()：为应用程序提供创建过滤器图表的方法；

IMediaControl()：提供控制过滤器图表中多媒体数据流的方法，包括运行、暂停和停止；

IMediaEventEx()：继承自 IMediaEvent 接口，处理过滤图表的事件；

IVideoWindow()：用于设置多媒体播放器窗口的属性，应用程序可以用它来设置窗口的所有者、位置和尺寸等属性；

IBasicAudio()：用于控制音频流的音量和平衡；

IBasicVideo()：用于设置视频特性，如视频显示的目的区域和源区域；

IMediaSeeking()：提供搜索数据流位置和设置播放速率的方法；

IMediaPositon()：用于寻找数据流的位置；

IVideoFrameStep()：用于步进播放视频流，可使 DirectShow 应用程序，包括 DVD 播放器一次只播放一帧视频。

4.2.1.5 捕捉图表生成器（Capture Graph Builder）

捕捉图表生成器为创建过滤器图表提供了附加的方法。最初，它是为构建视频捕捉图表而被设计出来的，但对于其它类型的过滤器图表也很有作用。它提供 ICaptureGraphBuilder2 接口。

ICaptureGraphBuilder2 接口是处理视频捕捉工作的接口，它提供了一个 filter graph builder 对象，让应用程序在建立捕捉过滤器图表时，省去处理很多单调乏味的工作，集中精力于捕捉中。它提供的方法满足了基本的捕捉和预览功能的要求。

FindInterface()方法：在过滤器图表中查找一个与捕捉有关的详细的接口。使得用户可以访问一个详细接口的功能，而不需要你去列举过滤器图表中的 pins 和 filters。

RenderStream()方法：连接源过滤器和提交过滤器，选择添加一些中间的过滤器。

ControlStream()方法：独立精确地控制图的开始和结束帧。

4.2.1.6 过滤器图表中的数据流动

当用户要创建自定义的过滤器时，就需要了解媒体数据是如何在过滤器表中传输的。为了在过滤器图表中传送媒体数据，DirectShow 过滤器需要支持一些协议，称之为传输协议（transport）。相连的过滤器必须支持同样的传输协议，否则不能交换媒体数据。

大多数的 DirectShow 过滤器把媒体数据保存在主存储器中，并通过引脚把数据提交给其他过滤器，这种传输称为局部存储器传输（local memory transport）。

虽然局部存储器传输在 DirectShow 中最常用，但并不是所有的过滤器都使用它。例如，有些过滤器通过硬件传送媒体数据，引脚只是用来提交控制信息，如 IOverlay 接口。

DirectShow 为局部存储器传输定义了两种机制：推模式（push model）和拉模式（pull model）。在推模式中，源过滤器生成数据并提交给下一级过滤器。下一级过滤器被动地接受数据，完成处理后再传送给下一级过滤器。在拉模式中，源过滤器与一个分析过滤器相连。分析过滤器向源过滤器请求数据后，源过滤器才传送数据以相应请求。推模式使用的是 IMemInputPin 接口，拉模式使用 IAsyncReader 接口，推模式比拉模式要更常用。

4.2.1.7 系统设备枚举器（System Device Enumerator）

系统设备枚举器为按类型枚举已注册在系统中的 Filter 提供了统一的方法。而其它能够区分不同的硬件设备，即便是同一个 Fileter 支持它们。利用系统设备枚举器查询设备的时候，系统设备枚举器为特定类型的设备（如视频捕捉的视频压缩）生成了一张枚举表（Enumerator）。类型枚举器（Category enumerator）为每个这种类型的设备返回一个 Moniker，类型枚举器自动把每一种即插即用的设备包含在内。使用系统设备枚举器的步骤如下：

（1）调用方法 CoCreateInstance()生成系统设备枚举器；

（2）调用方法 ICreateDevEnum：CreateClassEnumerator()方法生成类型枚举器，该方法返回一个 IEnumMoniker 接口指针；

（3）使用 IEnumMoniker：Next()方法依次得到 IEnumMoniker 指针中的每个 moniker。该方法返回一个 IMoniker 接口指针；

（4）想要得到该设备较为友好的名称（例如想要在用户界面中进行显示），调用 IMoniker::BindToStorage()方法；

（5）如果想要生成并初始化管理该设备的 Filter，调用 3 返回指针的 IMoniker::BindToObject()，接下来调用 IFilterGraph::AddFilter()把该 Filter 添加到图表中。

如图 4.10 所示说明了上面的步骤。

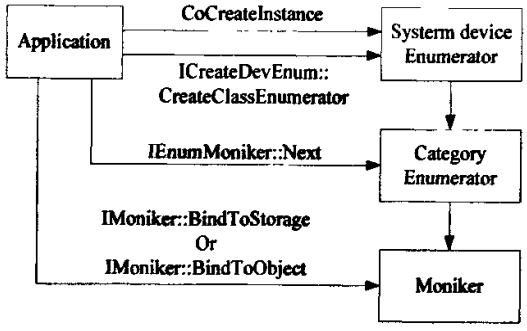


图 4.10 系统设备枚举器使用步骤

4.2.2 视频图像播放的相关程序说明及实现

利用 DirectShow 开发包所提供的 COM 接口，设计了一个类 CVideoPlay，来具体实现实时图像和视频文件的播放和操作功能。它提供的函数有：

FindDevice()：搜索视频设备；

RealPlay()：实时图像的播放；

PlayFromFile()：视频文件播放；

PausePlay()：暂停播放；

ResumePlay()：继续播放；

StopPlay()：停止播放；

DisplayVideoWin()：显示视频播放窗口；

下面将对各个函数的实现做详细说明。

4.2.2.1 FindDevice()函数

FindDevice()实现对当前系统存在的视频设备进行搜索，并返回设备名列表。它通过 DirectShow 的系统枚举器实现。首先创建一个 ICreateDevEnum 接口；在调用 ICreateDevEnum::CreateClassEnumerator()方法生成类型枚举器，返回一个 IEnumMoniker 接口；调用 IEnumMoniker::Next()方法的枚举每个视频设备，该方法返回一个 IMoniker 接口指针；然后通过调用 IMoniker::BindToStorage()方法得到视频设备的友好名称；最后将设备名添加到设备名列表 DevName 中返回。

```
//搜索视频设备，并返回视频设备名列表；
void CVideoPlay::FindDevice(CStringList &DevName)
```

```
    {
        HRESULT hr;
        int uIndex = 0;
        //创建一个系统设备枚举器接口;
        ICreateDevEnum *pCreateDevEnum;
        hr = CoCreateInstance(CLSID_SystemDeviceEnum, NULL,
    CLSCTX_INPROC_SERVER, IID_ICreateDevEnum, (void
    **)&pCreateDevEnum);
        //创建一个类型枚举器，指向系统的视频设备列表;
        IEnumMoniker *pEm;
        hr = pCreateDevEnum->
                CreateClassEnumerator(CLSID_VideoInputDeviceCategory, &pEm,
0);
        HELPER_RELEASE(pCreateDevEnum);
        if(pEm)
        {
            pEm->Reset();
            ULONG cFetched;
            IMoniker *pM;
            While(hr = pEm->Next(1, &pM, &cFetched), hr == S_OK)
            {
                IPropertyBag *pBag;
                hr = pM->BindToStorage(0, 0, IID_IPropertyBag, (void
**)&pBag);
                if(SUCCEEDED(hr))
                {
                    VARIANT var;
                    var.vt = VT_BSTR;
                    //得到视频设备的友好界面;
                    hr = pBag->Read(L"FriendlyName", &var, NULL);
                    if(hr == NOERROR)
                    {
                        CString achName;
                        wideCharToMultiByte(CP_ACP, 0, var.bstrVal, -1,
                        achName.GetBuffer(50), 80, NULL, NULL);
                        achName.ReleaseBuffer();
                        //将设备名添加到设备名列表末尾
                        DevName.AddTail(achName);
                        SysFreeString(var.bstrVal);
                    }
```

```
                HELPER_RELEASE(pBag);
            }
            HELPER_RELEASE(pM);
            uIndex++;
        }
        HELPER_RELEASE(pEm);
    }
}
```

4.2.2.2 RealPlay()函数

RealPlay()函数实现对实施捕捉图像的播放。它首先创建ICaptureGraphBuilder接口和IGraphBuilder接口，通过调用ICaptureGraphBuilder::SetFiltergraph()方法将捕捉图表和过滤器图表进行关联；再通过系统设备枚举器枚举系统中的每个视频设备，调用IMonitor::BindToObject()方法为每个设备生成一个过滤器管理该设备，并通过调用IFilterGraph::AddFilter()的方法将过滤器添加到过滤器图表中；然后调用ICaptureGraphBuilder::RenderStream()方法连接源过滤器和提交过滤器，再调用ICaptureGraphBuilder::FindInterface()方法查找与视频捕捉有关的现实窗口，最后调用IMediaControl::Run()方法实时播放捕捉视频流。

```
//实时捕捉图像的播放;
void  CVideoPlay::RealPlay()
{
    HRESULT hr;
    //创建捕捉列表;
    CHECK_ERROR(CoCreateInstance((REFCLSID)CLSID_CaptrueGraphBu
ilder, NULL, CLSCTX_INPROC, (REFIID)IID_ICaptureCraphBuilder, (void **)
&CapPigb), "CoCreateInstance Error");
    //创建过滤器图表;
    CHECK_ERROR(hr = CoCreateInstance(CLSID_FilterGraph, NULL,
CLSCTX_INPROC, IID_IGraphBuilder, (LPVOID *)&CappFg), "Cannot instantiate
filtergraph");
    //将捕捉图表和过滤器图表进行关联;
    hr = CapPigb->SetFiltergraph(CappFg);
    if(hr != NOERROR)
    {
        MessageBox(m_hwnd, "Cannot give graph to builder", "Error",
```

```
MB_OK);
            return;
        }
        int uIndex = 0;
        //创建视频设备枚举器;
        ICreateDevEnum *pCreateDevEnum;
        hr      =      CoCreateInstance(CLSID_SystemDeviceEnum,      NULL,
CLSCTX_INPROC_SERVER, IID_ICreateDevEnum, (void **)&pCreateDevEnum);
        //创建一个类型枚举器，指向系统的视频设备列表;
        IEnumMoniker *pEm;
        hr = pCreateDevEnum->
                CreateClassEnumerator(CLSID_VideoInputDeviceCategory, &pEm,
            0);
        HELPER_RELEASE(pCreateDevEnum);
        pEm->Reset();
        ULONG cFetched;
        IMoniker *PM;
        /枚举每个视频设备;
        while(hr = pEm->Next(1, &pM, &cFetched,), hr == S_OK)
        {
            //生成并初始化管理该设备的过滤器;
            hr = pM->BindToObject(0, 0, IID_IBaseFilter, (void **)&pVCap);
            if(pVCap == NULL)
            {
                MessageBox(m_hwnd, "Cannot get the capture filter", "Error",
MB_OK);
                return;
            }
            HELPER_RELEASE(pM);
            uIndex++;
        }
        HELPER_RELEASE(pEm);
        if(pVCap)
            hr = CappFg->AddFilter(pVCap, NULL);  //添加过滤器到 Filter
Graph 中
        if(hr != NOERROR)
        {
            MessageBox(m_hwnd, "Cannot add vidcap to filtergraph", "Error",
MB_OK);
            return;
```

```
        }
        //连接源过滤器和提交过滤器;
        hr = CapPigb->RenderStream(&PIN_CATEGORY_PREVIEW, pVCap,
    NULL, NILL);
        if(hr != S_OK)
        {
            MessageBox(m_hwnd, "This graph cannot preview properly", "Error",
MB_OK);
            return;
     }
        //在过滤器图表中查找一个与捕捉有关的接口;
        hr = CapPigb->RindInterface(&PIN_CATERGORY_PREVIEW, pVCap,
    IID_IVideoWindow, (void **)&pivw);
        if(hr != ERROR)
        {
            MessageBox(m_hwnd, "cannot find Video Window properly", "Error",
MB_OK);
            return;
        }
        else
            DisplayVideoWin();          //显示视频播放窗口;
        //查询数据流控制接口;
        hr = CappFg->QueryInterface(IID_IMediaControl, (void **)&pimc);
        if(SUCCESSED(hr))
            hr = pimc->Rum();
       else
            MessageBox(m_hwnd, "Cannot run preview graph", "Error", MB_OK);
    }
```

4.2.2.3 PlayFromFile()函数

PlayFromFile()函数实现对视频文件的播放。它首先创建一个 IGraphBuilder 接口，调用 IGraphBuilder::RenderFile()方法为多媒体文件创建一个过滤器图表进行处理；然后调用 IMediaControl::Run()方法播放视频。

4.2.2.4 PausePlay()函数和 ResumePlay()函数

PausePlay()函数暂停视频的播放。它通过调用 IMediaControl 接口的 Pause()方法实现。ResumePlay()函数继续视频的播放。它首先判断当前的文件位置，如果已在播放文件的最后（播放时间剩下不到 1 分钟），则将当前位置为文件开始

处；然后调用 IMediaControl 接口的 Run()方法从当前位置开始播放。

4.2.2.6 StopPlay()函数

StopPlay()函数停止视频的播放。它首先调用 IMediaControl 接口的 stop()方法停止视频的播放，然后关闭视频播放窗口，并释放相应的接口。

4.2.2.7 DisplayVideoWin()函数

DisplayVideoWin()函数显示视频播放窗口。它通过调用 IVideoWindow 接口的有关方法实现。

4.3 视频数据的有线传输和接受播放

监控现场主机通过视频数据发送模块，将现场采集到的视频流数据以 IP 组播的形式通过计算机网络发送出去。对于发送过来的视频数据，运行在客户端的视频数据接收播放模块一方面可以将其保存起来，构成资料库便于以后查询及回放；另一方面还可以实时播放出来，使远程现场的监控情景呈现在用户面前，达到远程监控的目的。这部分主要涉及到 Winsock 网络编程和 Windows 多线程编程技术，下面先将对它们作一介绍。

4.3.1 Winsock 网络编程技术

4.3.1.1 Winsock 的基本概念

Socket 即套结字，最初是由 U.C.Berkeley 为 UNIX 操作系统开发的网络通信接口。Socket 是网络通信的基本构件，一个 Socket 对应于通信的一端。网络通信的 Socket 接口模型将通信主机当作端点。每个网络通信对应两个端点：本地主机和远地主机。Socket 接口将网络通信的每个端点称之为一个 Socket。Socket 是可以被命名和寻址的通信端点，一个正在被使用的 Socket 都有它的类型和与其相关的进程。

网络进程通信的首要问题是进程标识。在同一台主机中，不同进程可以用进程号唯一标识，但在网络环境中，各主机独立分配的进程号是不能作为进程标识。另一方面，网络中进程本身也不用进程号来描述，一个比进程更低级、更稳定的概念是端口，端口是 TCP/IP 与应用程序打交道的访问点，是 TCP/IP

协议的一部分。一个主机内的每个网络进程使用协议端口进行标识。这样，要唯一确定网络环境下的某个进程，就同时需要 IP 地址和端口号。网络通信需要解决的第二个问题是多重协议的标识。网络进程间通信时，必须在众多的通信协议中指明是何种通信协议。

综上所述，在网络环境中全局地标识一个进程需要一个三元组：协议、本地地址、本地端口号。这样一个三元组成为一个半相关。而一个完整的网络进程间通信需要一个五元组来标识：协议、本地地址、本地端口号、远地地址、远地端口号。这样的一个五元组成为一个相关。

WinSock 是 Windows 下提供的网络编程接口，它是从 UNIX 下的 Berkeley Socket 扩展而来的。它在继承了 Berkeley Socket 主要特征的基础上，又对它进行了重要的扩充。WinSock 规范把 API 划分为以下 3 个部分：

（1）与 Berkeley Socket 兼容的基本 Socket 函数，用来创建、打开、关闭 Socket，设置 Socket 属性、发送接受数据等；

（2）网络数据信息检索函数，应用程序用它取回域名、通信服务、协议等网络信息；

（3）Winsock 专有的为 Windows 专用的扩展函数。

4.3.1.2 Winsock 的编程特点

在网络通信中，由于网络拥挤或一次发送的数据量过大等原因，经常会发生交换的数据在短时间内不能传送完，收发数据的函数因此不能返回，这种现象叫做阻塞。Winsock 对有可能阻塞的函数提供了两种处理方式：阻塞方式和非阻塞方式。在阻塞方式下，收发数据的函数在被调用后一直要到传送完毕或者出错才能返回。在阻塞期间，除了等待网络操作的完成不能进行任何操作。对于非阻塞方式，函数被调用后立即返回，当网络操作传送完成后由 Winsock 给应用程序发送一个消息通知操作完成，此时可以根据发送的消息传出的参数判断操作是否正常。在编程时，应尽量使用非阻塞方式。因为在阻塞方式下，一方面会因为某个函数的阻塞而使别的进程无法进行，一个进程的长期挂起会造成系统的死机；另一方面，用户可能会因为长时间的等待而失去耐心继而关闭应用程序的主窗口，这样当网络操作的函数从 Winsock 的动态连接库中返回时，此时主程序已经从内存中删除，可能会造成内存的异常。

4.3.1.3 客户机/服务器模型的 WinSock 程序设计

在 TCP/IP 网络应用中，通信的两个进程间相互作用的主要模式是客户/服务器模式。网络进程间通信所遵循的协议有两种：一种是可靠的、面向连接的、数据流的 TCP 协议，另一种是不可靠的、无连接的、数据的 UDP 协议。针对这两种通信方式，WinSock 程序设计有两种对应的流程：

（1）面向连接协议 TCP 的 Winsock 程序设计流程图

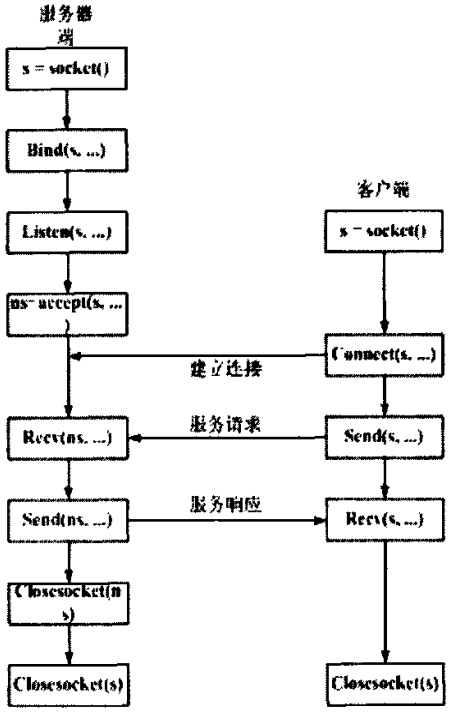


图 4.11 面向连接的 Winsock 程序设计流程图

如图 4.11 所示是面向连接的 Winsock 程序设计流程图。在面向连接的协议中，服务器和客户机开始通信之前必须首先建立连接。在连接之前，服务器程序必须正在运行并处于侦听模式，等待客户机的连接。为了使服务器程序处于运行状态和倾听模式，服务器程序必须完成几项操作，如流程图所示，首先必须创建一个套接字 s，即 s=socket()，其次必须把该套接字和一个 IP 地址邦定在一起，即 bind(s, ...)。此时服务器便进入了侦听模式，等待客户机往该套接字上连接，即 listen(s, ...)。当客户机连向服务器时，也建立一个套接字，即 s=socket()，

但客户机并没有用到函数 bind(),这是因为 TCP/IP 网络上的面向连接客户程序并不关心客户端使用什么样的本地地址进行数据传输，因此不需要用 bind 函数将客户 Socket 与本地 IP 地址邦定在一起。客户机建立 Socket 以后开始尝试与服务器套接字的 IP 地址连接，即 connect(s, ...)。Connect 函数启动和远地主机的直接连接，即在连接端点之间建立一条虚电路。一旦服务器程序检测到客户机与它连接后，就创建一个新的套接字，这个套接字用来与客户机进行通信，即 ns=accept(s, ...)，ns 就是新的套接字，而后服务器的原始套接字 s 转去倾听其他客户机的连接。这样，与服务器套接字邦定的 IP 地址总是保留给其他客户机使用。后续客户机的每一个连接都得到一个唯一的服务器套接字，以进行通讯。所以，一个服务器能在同一时间与多个客户机进行通信。在建立了连接之后，服务器和客户机就可以开始通信，即分别用 recv()和 send()接受和发送数据，recv()和 send()函数分别用于通过连接的 Socket 接收和发送数据。通信结束后，服务器和客户机各调用 closesocket()关闭所有套接字，并释放套接字的资源。

（2）面向无连接协议 UDP 的 Winsock 程序设计流程

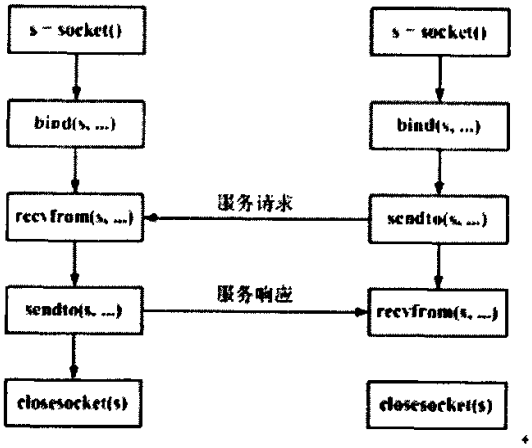


图 4.12 面向无连接的 Winsock 程序设计流程

如图 4.12 所示是面向无连接的 Winsock 程序设计流程图。从流程图中可以看到，面向无连接模式与面向连接模式的最大区别在于 Socket 在两台主机进行通讯之前无须在两个端点之间建立点到点连接，即不用函数 connect()来建立连接，这就是所谓的数据报 Socket。但这时需要在客户端运用 bind 函数，这是因为无连接协议的客户程序在发送数据时必须包含客户端的本地地址参数，这就

是异步通信所必需的。在面向连接的模式中，数据的请求和应答是在 Socket 建立连接之后同步进行的，即在请求的同时必定有应答。在这种模式中，函数 send()和 recv()的功能就是通过一个连接的 Socket 来传送和接收数据。而在面向无连接的模式中，数据的请求和应答是异步进行的，与连接模式相对的是函数 sendto()和 recvfrom()可以在无连接的 Socket 中完成数据传送和接收的功能，这是因为在这两个函数中可以包含“本地地址”和“目的地址”的参数信息，从而不必在连接之后同步传送数据，而可以异步进行。

4.3.1.4 Winsock 的异步模式

Winsock 通过异步选择函数 WSAAsyncSelect()来实现非阻塞通信。方法是由该函数制定某种网络事件（如：由数据到达，可以发送数据，有程序请求连接等），当被指定的网络事件发生时，由 Winsock 对程序发送由程序实现约定的消息。程序中就可以根据这些消息做相应的处理。格式如下：int WSAAsyncSelect (SOCKET s, HWND hwnd, unsigned int wMsg, long iEvent);

Sockets 在这个函数调用中被自动设置成非阻塞方式，hWnd 是接受 Winsock 消息的窗口句柄，非阻塞方式下 wMsg 是向窗口发出消息名称，用户可以任意定义。lEvent 是被指定的网络事件，有下面几种选择：

表 4.1 网络事件的值及所代表的含义

值	代表的含义
FD_READ	希望 Socket 收到数据时发出读的消息
FD_WRITE	希望 Socket 发送数据时发送写的消息
FD_OOB	希望 OOB data 到达时发送到达的消息
FD_ACCEPT	希望有连接到来的时候发送连接请求的消息
FD_CONNECT	希望完成连接时发送连接完成的消息
FD_CLOSE	希望接受 Socket 关闭的信息
FD_QOS	希望接受 Socket 服务质量（QOS）变化的消息
FD_GROUP_QOS	希望接受 Socket 服务质量不变的消息
FD_ROUTING_INTERFACE_CHANGE	希望接受制定的地址路由接口变化的消息
FD_ADDRESS_LIST_CHANGE	希望接受 Socket 协议族局部地址变化的消息

4.3.1.5 Winsock 多址广播

多址广播是一种一对多的传输方式，传输发起者通过一次传输就将信息传送到一组接受者，与单点传送和广播相对应。多址广播使用最广泛的是 IP 组播，它是标准 IP 网络层协议的扩展。

Winsock2 为支持 IP 组播而定义了一组新的与协议无关的多址广播应用程序接口，归纳起来可以用表 4.2 表示。

表 4.2 Winsock2 的多址广播 API

WSAEnumProtocols()	检测多址广播支持
WSASocket()	制定多址广播类型
WSAJoinLeaf()	加入一个多址广播组并制定角色（发送者和/或接收者）
WSAIoctl() SIO_MULTICAST_SCOPE	设置 IP 生存时间
WSAIoctl() SIO_MULTIPOINT_LOOPBACK	禁止内部回送（loopback）

函数 WSAEnumProtocols()返回当前系统中安装的协议的详细描述，这些信息存放在一个协议信息结构（WSAPROTOCOL_INFO）的数组中。在其中的域 dwServiceFlags1 中的一些标识指示此服务有 IP/UDP 协议提供，并且位标识 XP1_SUPPORT_MULTIPOINT 指示该服务支持 IP 组播。

为了适应不同的多址广播模式，Winsock2 定义了数据平面（data plane）和控制平面（control plane）两个概念，每一个平面都可以是“有根（rooted）”的或“无根（non-rooted）”的。IP 组播是一种无根的数据平面和控制平面。在使用 WSASocket()函数请求一个多址广播套接字时需要指定这些角色。这通过在 WSASocket()函数的参数 dwFlags 中使用四个标志位来实现：

WSA_FALG_MULTIPOINT_C_ROOT，用来创建一个作为 c_root 节点的套接字，并且只有在相应的 WSAPROTOCOL_INFO 入口中指示了使用 rooted 控制平面时才允许。

WSA_FALG_MULTIPOINT_C_LEAF，用来创建一个作为 c_leaf 节点的套接字，并且只用在相应的 WSAPROTOCAL_INFO 入口中指示了 XP1_SUPPORT_MULTIPOINT 时才允许。

WSA_FALG_MULTIPOINT_D_ROOT，用来创建一个作为 d_root 节点的套接字，并且只有在相应的 WSAPROTOCOL_INFO 入口中指示了使用 rooted 数据平面时才允许。

WSA_FALG_MULTIPOINT_D_LEAF，用来创建一个作为 d_leaf 节点的套接字，并且只有在相应的 WSAPROTOCOL_INFO 入口中指示了 XP1_SUPPORT_MULTIPOINT 时才允许。

* 注意：在 IP 组播中，只有标志 WSA_FLAG_MULTIPOINT_C_LEAF 和 WSA_FLAG_MULTIPOINT_D_LEAF 能用来作为 WSASocket()函数的 dwFlags 参数。

WSAIoctl()函数的 SIO_MULTIPOINT_LOOPBACK 命令码用来设置是否允许内部回送。当 d_leaf 套接字用于 non-rooted 数据平面时，它通常是希望能够控制发送出去的通信流量是否能够在同一个套接字上也被接受。WSAIcotl()函数的 SIO_MULTIPOINT_LOOPBACK 命令码用来允许或禁止多址广播的通信流量的内部回送。

WSAIoctl()函数的 SIO_MULTICAST_SCOPE 命令码用来设置多址广播范围。当使用多址广播是，常常需要制定多址广播传播的范围。范围由包括的路由网段来定义。范围为 0 指示多址广播不传播信息到“网线”上，但是可以在本地主机的多个套接字间传播。范围为 1（默认值）指示将传播信息到“网线”上，但是不跨越路由器。更高的范围值决定了可以跨越的路由器数量。

WSAJoinLeaf()函数用来加入一个叶子节点到多址广播会话，其函数原形为：

```
SOCKET WSAAPT WSAJoinLeaf( SOCKET s,
const struct sockaddr FAR *name,
int namelen,
LPWSABUF lpCallerData,
LPWSABUF lpCalleeData,
LPQOS lpSQOS,
LPQOS lpGQOS,
DWORD dwFlags
);
```

WSAJoinLeaf()具有与 WSAConnect()相同的参数和语法，除了它还返回一个套接字描述符（这和函数 WSAAccept()一样）及多了一个 dwFlags 参数外。参数 dwFlags 用来指示套接字是用来作为发送者还是接受者还是两者兼具。在此函数中，只有多址广播套接字可以用来作为输入参数 s。如果此多址广播套接字处于非阻塞模式，返回的套接字描述符只有在接收到相应的 FD_CONNECT 指示后

才能使用。参数 name 指示套接字将加入的多址广播的地址。

另外，在上面的 API 中没有提到如何离开一个多址广播址。惟一与协议无关的 API 是关闭套接字，即使用标准的 closesocket()函数，它可以用来离开多址广播组。

4.3.2 Windows 多线程编程技术

Windows 是一种多任务的操作系统，在 Windows 的一个进程内包含一个或多个线程。32 位 Windows 环境下的 Win32 API 提供了多线程应用程序开发所需要的接口函数，而利用了 VC 中提供了的标准 C 库已可以开发多线程应用程序，相应的 MFC 类库封装了多线程编程的类，用户在开发时可根据应用程序的需要和特点选择相应的工具。下面将重点介绍 Win32 API 和 MFC 两种方式下如何编制多线程程序。

多线程编程在 Win32 方式下和 MFC 下的原理是一致的，进程的主线程在任何需要的时候都可以创建新的线程。当线程执行完后，自动终止线程；当进程结束后，所有的线程都终止。所有活动的线程共享进程的资源，因此，在编程时需要考虑在多个线程访问同一资源时产生冲突的问题。当一个线程正在访问某进程对象，而另一个线程要改变该对象，就可能会产生错误的结果，编程时要解决这个冲突。

4.3.2.1 Win32 API 下的多线程编程

Win32 API 是 Windows 操作系统内核与应用程序之间的界面，它将内核提供的功能进行函数包装，应用程序通过调用相关函数而获得相应的系统功能。为了向应应用程序提供多线程功能，Win32 API 函数集中提供了一些处理多线程程序的函数集。直接用 Win32 API 进行程序设计具有很多优点，基于 Win32 的应用程序执行代码小，运行效率高，但是它要求程序员编写的代码较多，且需要管理所有系统提供给程序的资源。

（1）创建和终止程序

Win32 函数库中提供了操作多线程的函数，包括创建线程、终止线程、建立互斥区等。在应用程序的主线程或者其他活动线程中创建新的线程的函数如下：

```
HANDLE CreateThread(LPSECURITY_ATTRIBUTES lpThreadAttributes,
DWORD dwStackSize,
```

LPTHREAD_START_ROUTINE lpStartAddress,

LPVOID lpParameter,

DWORD dwCreationFlags,

LPDWORD lpThreadId);

如果创建成功则返回线程的句柄，否则返回 NULL。创建了新的线程后，该线程就开始启动执行了。但如果在 dwCreationFlags 中使用了 CREATE_SUSPENDED 特性，那么线程并不马上执行，而是先挂起，等到调用 ResumeThread 后才开始启动线程，在这个过程中可以调用下面这个函数来设置线程的优先权：

BOOL SetThreadPriority(HANDLE hThread, int nPriority);

当调用线程的函数返回后，线程自动终止。如果需要在线程的执行过程中终止则可调用函数：

VOID ExitThread(DWORD dwExitCode);

如果在线程的外面终止线程，则可调用下面的函数：

BOOL TerminateThread(HANDLE hThread, DWORD dwExitCode);

如果要终止的线程是进程内的最后一个线程，则线程被终止后，相应的进程也应终止。

（2）线程同步

在线程体内，如果该线程完全独立，与其他线程没有数据存取等资源操作上的冲突，则可按照通常单线程的方法进行编程。但是，在多线程处理时情况常常不是这样，线程之间经常要同时访问一些资源。由于对共享资源进行访问引起冲突是不可避免的，为了解决这种线程同步问题，Win32 API 提供了多种同步控制对象来帮助程序员解决共享资源访问冲突。在介绍这些同步对象之前先介绍一下等待函数，因为所有控制对象的访问控制都要用到这个函数。

Win32 API 提供了一组能使线程阻塞其自身执行的等待函数。这些函数在其参数中的一个或多个同步对象产生了信号，或者超过了规定的等待时间才会返回。在等待函数未返回时，线程处于等待状态，此时线程只消耗很少的 CPU 时间。使用等待函数既可以保证线程的同步，又可以提高程序的运行效率。最常用的等待函数是：

DWORD WaitForSingleObject(HANDLE hHandle, DWORD dwMilliseconds);

而函数 WantForMultipleObject 可以用来同时监测多个同步对象，该函数的

声明为：

DWORD WaitForMultipleObject(DWORD nCount, CONST HANDLE *lpHandles, BOOL bWaitAll, DWORD dwMilliseconds);

下面看一下各种同步对象。①互斥体对象（Mutex）对象的状态在它不被任何线程拥有时才有信号，而当它被拥有时则无信号。Mutex 对象很适合用来协调多个线程对共享资源的互斥访问。②信号对象（Semaphore）允许同时对多个线程共享资源进行访问，在创建对象时指定最大可同时访问的线程数。当一个线程申请访问成功后，信号对象中的计数器减一，调用 ReleaseSemaphore()函数后，信号对象中的计数器加一。其中，计数器值大于或等于 0，但小于或等于创建时指定的最大值。如果一个应用在创建一个信号对象时，将其计数器的初识值设为 0，就阻塞了其它线程，保护了资源。等初始化完成后，调用 ReleaseSemaphore()函数将其计数器增加至最大值，则可进行正常的存取访问。③事件对象（Event）是最简单的同步对象，它包括有信号和无信号两种状态。在线程访问某一资源之前，需要等待某一事件的发生，这时用事件对象最合适。④排斥区对象（CriticalSection），在排斥区中异步执行时，它只能在同一进程的线程之间共享资源处理。虽然此时上面介绍的几种方法均可使用，但是，是用排斥区的方法则使同步管理的效率更高。

4.3.2.2 MFC 下的多线程编程

在 MFC 中，线程分为两种：工作线程和用户界面线程（UI）线程。这种区分是由 MFC 自己进行的，Win32 API 不区分线程的种类。

（1）工作线程

工作线程一般用来完成那些不需要用户输入的后台任务，因此他没有窗口和消息循环。工作线程编程较为简单，设计思路与前面所讲的基本一致：一个基本函数代表了一个线程，创建并启动线程后，线程进入运行状态；如果线程用到共享资源，则需要进行资源同步处理。

创建线程并启动线程时调用 AfxBeginThread。函数原形说明如下：

```
CWinThread* AfxBeginThread(AFX_THREADPROC pfnThreadProc,
LPVOID pParam,
Int nPriority = THREAD_PRIORITY_NOMAL,
UINT nStackSize = 0,
```

DWORD dwCreateFlags = 0,

LPSECURITY_ATTRIBUTES lpSecurityAttrs = NULL);

参数 pfnThreadProc 是线程执行体函数，函数原形为：

UINT ThreadFunctoin(LPVOID pParam);

参数 pParam 是传递给执行函数的参数。

参数 nPriority 是线程执行权限，可选择：THREAD_PRIORITY_NOMAL、THREAD_PRIORITY_LOWEST 、 THREAD_PRIORITY_HIGHEST 、THREAD_PRIORITY_IDLE。

参数 dwCreateFlags 是线程创建时的标志，可取值 CREATE_SUSPENDED，表示线程创建后处于挂起状态，调用 ResumeThread 函数后线程继续运行，或者取值“0”表示线程创建后处于运行状态。

返回值是 CWinThread 类对象指针，它的成员变量 m_hThread 为线程句柄，在 Win32 API 方式下对线程操作的函数参数都要求提供线程的句柄，所以当线程创建后可以使用所有 Win32 API 函数对 pWinThread->m_Thread 线程进行相关操作。

（2）用户界面线程

用户界面线程能够处理用户输入，它们通过实现消息循环来响应那些有用户与应用程序交互产生的事件和消息。基于 MFC 的应用程序有一个应用对象，它是 CWinApp 派生类的对象，该对象代表了应用程序的主进程。当线程执行完并退出线程时，由于进程中没有其他线程存在，进城自动结束。类 CWinApp 从 CWinThread 派生出来，CWinThread 是用户界面线程的基本类。在编写用户界面线程时，需要从 CWinThread 派生自己的线程类，ClassWizard 可以帮助我们完成这个工作。

先用 ClassWizard 派生一个新的类，设置基类为 CWinThread。注意：类的 DECLARE_DYNCREATE 和 IMPLEMENT_DYNCREATE 宏是必需的，因为创建线程时需要动态创建类的对象。根据需要可将初始化和结束代码分别放在类的 InitInstance()和 ExitInstance()函数中。如果需要创建窗口，则可在 InitInstance()函数中完成。然后创建线程并启动线程。可以用两种方法来创建用户界面线程，MFC 提供了两个版本的 AfxBeginThread()函数，其中一个用于创建用户界面线程。第二种方法分为两步进行：首先，调用线程类的构造函数创建一个线程对象；其次，调用 CWinThread::CreateThread()函数来创建该线程。线程建立并启

动后，在线程函数执行过程中一直有效。如果是线程对象，则在对象删除之前，先结束线程。CWinThread 已经为我们完成了线程结束的工作。

（3）线程同步

前面介绍了 Win32 API 提供的几种有关线程同步的对象，在 MFC 类库中对这几个对象进行了封装，它们有一个共同的基类 CSyncObject，它们的对应关系为：Semaphore 对应 CSemaphore、Mutex 对应 CMutex、Event 对应的 CEvent、CriticalSection 对应 CCriticalSection。另外，MFC 对应两个等待函数也进行了封装，即 CSingleLock()和 CMultiLock()。

这 4 个对象分别适用于不同的场合：如果某个线程必须等待某些事件发生后才能存取相应的资源，则用 CEvent；如果一个应用同时可以有多个线程存取相应资源，则用 CSemaphore；如果有多个应用（多个进程）同时存取相应资源，则用 CMutex，否则用 CCriticalSection。

4.3.3 视频数据发送和接收播放模块设计

视频数据发送模块运行在监控现场主机，它以组播的方式发送视频流；视频数据接受播放模块运行在客户端主机，该部分接受视频流并实施播放显示。考虑到监控中心（客户端）主机要同时进行监控若干现场，需要加入多个组播组的情况，在设计系统时要求监控中心在接受视频数据前应向相应的监控现场发送数据请求。因此，要建立两个通信通道：一个是控制通道，一个为数据通道。控制通道用来在发送端和接受端之间建立会话，包括发送一些数据请求和确认控制等信息。接受端在接受视频数据前先向相应的发送端发送数据请求，发送端在接受到数据请求后向接受端发回相应的 IP 组播地址和端口，接受端则加入此组播接受视频流数据。为了保证这些控制信息准确无误地到达对方，对于控制通道应选择可靠性较高的 TCP 协议；数据通道用于视频流数据的通信，选用 IP 组播来实现，它是基于 UDP 协议的。这两个通信通道互不相关，各自执行自己的任务。

4.3.3.1 视频数据发送模块的设计

视频数据发送模块除了能发送实时监控视频流外，还能发送视频文件。无论是实时采集的视频数据还是从源视频文件获取的视频流，都可以在进行播放的同时发送数据，因此采用多线程技术。

视频文件发送的基本过程如下：选择打开一个视频文件；建立 Winsock2 Multicast Socket；启动文件发送线程，每次读文件 32KB 时由 Socket 发送出去。实时视频流的发送要比文件的发送复杂些，实时组播的视频流来自于视频采集压缩卡，所以首先要从视频采集压缩卡中获得视频数据。此外还应该设置一定的缓冲区来存放采集的视频数据，为此设计了一个视频流缓冲区，该视频流缓冲区的作用是设置一个缓冲区队列，对采集的数据进行数据的压入和弹出工作。从视频采集压缩卡采集的视频流数据不断地压入缓冲区队列，该缓冲区队列是有一定的大小限制，当缓冲区满时，则启动数据发送线程，将缓冲区的视频流数据弹出，以 IP 组播的方式发送出去，同时再将缓冲队列清空，以接受新的数据。重复上述过程，直到停止采集或视频发送过程。实时视频图像发送的基本过程如下。

（1）建立 Winsock2 Multicast Socket；

（2）初始化视频采集压缩卡，启动压缩采集；

（3）建立输出到缓冲区的流，由回调函数将采集的视频数据送至缓冲区，缓冲区的大小设为 32KB。每当缓冲区的数据满 32KB 时启动数据发送线程，通过 IP 组播方式将缓冲区的数据发送出去。

视频数据发送模块流程图如图 4.13 所示。

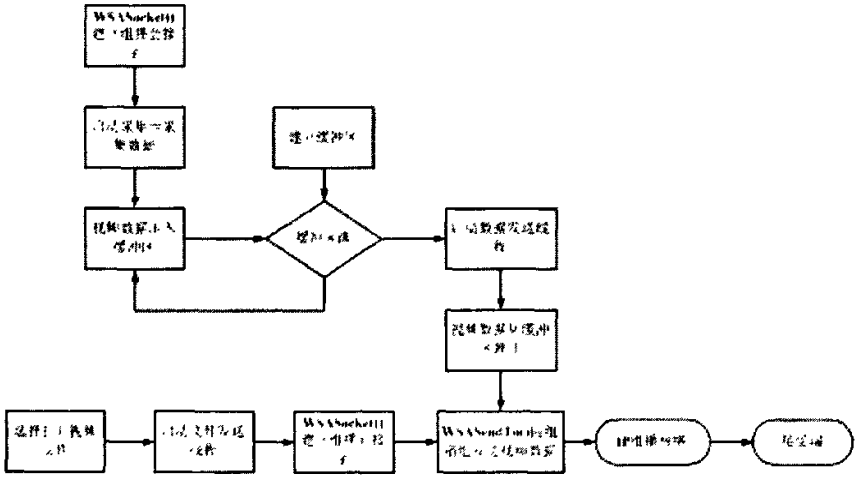


图 4.13 视频数据发送模块流程图

4.3.3.2 视频数据接受播放模块的设计

为了能同时接受播放多路图像，在程序的实现上采用了 Windows 用户界面线程技术，每个线程创建一个用户界面窗口，并负责一路视频流的接受和播放。

视频流的接受播放过程如下：

（1）输入连接发送端的 IP 地址，向发送端发送数据请求；

（2）当得到发送端的确认消息后，启动接受播放数据线程，创建用户窗口界面；

（3）建立 Winsock2 Multicast Socket，根据发送端传回的组播地址及端口号加入此 IP 组播组；

（4）建立 DirectShow Filter Graph，并启动运行；

（5）在 DirectShow 请求数据时，从 Socket 中读数据（每个 IP 数据包为 32K）送至 DirectShow 的 Buffer，此后视频流的解码和播放都由 DirectShow 来实现。

视频数据接受播放模块的流程图如图 4.14 所示。

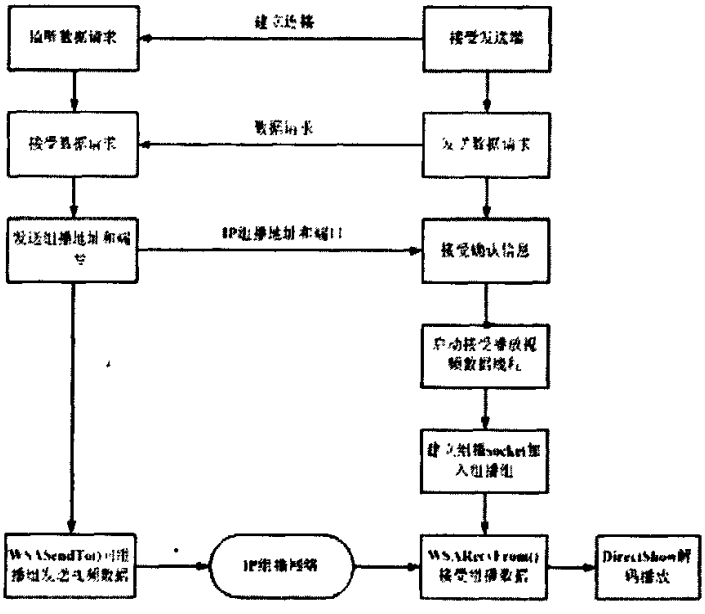


图 4.14 视频数据接受播放模块的流程图

4.3.4 视频数据发送模块的程序说明及实现

4.3.4.1 控制通道的实现

控制通道用来在发送端和接受端之间建立会话，包括发送一些数据请求和确认控制等信息。接收端在接收视频流数据前先向相应的发送端发送数据请求，发送端在接受到数据请求后向接收端发回相应的 IP 组播地址和端口，接收端则加入此组播组接受视频流数据。控制通道选择可靠性较高的 TCP 协议。

（1）InitSocket()函数

InitSocket()函数定义在“hikserver.ccp”文件中。它首先创建一个监听 socket，绑定发送端的主机地址和端口，然后监听接收端的连接请求，当有连接请求来到时，发送 WSA_ACCEPT 消息给主窗口。

```
//建立监听 socket
DWORD CPlayApp::InitSocket()
{
    SOCKADDR_IN send_sin;
    int status;
    CMainFrame* pFrame;
    pFrame=(CMainFrame*)m_pMainWnd;
    HWND m_hwndRec=pFrame->m_hWnd;
    pFrame->m_bAutoMenuEnable=FALSE;
    //创建一个 socket
    Lsock = socket(AF_INET,SOCK_STREAM,0);    //af 为网络地址类型，一
般//为 AF_INET，表示在 Internet 域中使用。
    if (Lsock == INVALID_SOCKET)
    {
        ErrMsg(m_hwndRec,"Socket failed");
        return -1;
    }
    send_sin.sin_port=htons(1500);
    send_sin.sin_family=AF_INET;
    send_sin.sin_addr.s_addr=INADDR_ANY;
    //绑定服务器主机地址与端口
    if (bind(Lsock, (struct sockaddr FAR *) &send_sin,
sizeof(send_sin))==SOCKET_ERROR)
    {
```

```
            ErrMsg(m_hwndRec,"bind failed");
            closesocket(Lsock );
            return -1;
        }
        //设置端口状态为监听
        if (listen(Lsock, 10) < 0)
        {
            ErrMsg(m_hwndRec,"Listen failed");
            closesocket(Lsock );
            return -1;
        }
        //设定服务器响应的网络事件为 FD_ACCEPT，即程序想要接收数据；
        //产生相应传递给窗口的消息为 WSA_ACCEPT
        if                                  ((status                                =
WSAAsyncSelect(Lsock,m_hwndRec,WSA_ACCEPT,FD_ACCEPT))>0)
        {
            ErrMsg(m_hwndRec,"Error on WSAAsyncSelect()");
            closesocket(Lsock );
            return -1;
        }
        return 0;
    }
```

（2）OnAccept()函数

OnAccept()函数定义在“HikVisionSdk.cpp”文件中。它响应 WSA_ACCEPT 消息，接受连接请求，与接收端建立连接。当有 FD_READ 或 FD_CLOSE 网络事件发生时，发送 WSA_READ 消息给主窗口。

```
    //响应消息 WSA_ACCEPT
    LRESULT CMainFrame::OnAccept(WPARAM wParam,LPARAM lParam)
    {
        int acsock;
        int status;
        if (WSAGETSELECTERROR( lParam ))
            return -1;
        if (WSAGETSELECTERROR( lParam ) == 0)
        {    /* Success */
            int req_sin_len = sizeof(req_sin );

            //接受客户的连接请求
```

```
            acsock = accept(Lsock,(struct sockaddr FAR *) &req_sin,(int FAR
*)&req_sin_len);
            if (acsock < 0)
            {
                MessageBox("Cant Accepted a connection!");
                return -1;
            }
            //设定服务器响应的网络事件为 FD_READ 或 FD_CLOSE，即读取
数据//或关闭 socket
            //产生相应传递给窗口的消息为 WSA_READ
            if ((status = WSAAsyncSelect(acsock, m_hWnd, WSA_READ,
FD_READ|FD_CLOSE))<0)
            {
                MessageBox("Error on WSAAsyncSelect()");
                closesocket(acsock );
                return -1;
            }
        }
        return 0;
    }
```

（3）OnRead()函数

OnRead()函数定义在“HikVisionSdk.cpp”文件中。它响应 WSA_READ 消息，对 FD_READ 或 FD_CLOSE 网络事件进行处理。FD_READ 表示发送端读取接收端传过来的数据，如果接收端传过来的是请求发送数据命令，则发送组播地址和端口给他；FD_CLOSE 表示接收端已经接收到组播地址信息，发送端可以关闭监听 socket。

```
//响应消息 WSA_READ
LRESULT CMainFrame::OnRead(WPARAM wParam,LPARAM lParam)
{
    int status;
    char szRev[80];
    char szBuff[80];
    char szSend[80];
    strcpy(szSend,MULTIDESTADDR);
    strcat(szSend, strDESTPORT);
    if (WSAGETSELECTERROR( lParam ))
        return -1;
```

```
        if (WSAGETSELECTEVENT(lParam) == FD_READ)
        {//网络事件为 FD_READ

            //接收数据
            status =recv(wParam, szRev, 80,0);
            if (status)
            {
                //如果客户端请求发送数据，将组播地址和端口发送给客户端
                if (strcmp(szRev,"请发送数据")==0)
                {
                    sprintf(szBuff," 来 自 %s 请 求 数 据 ", inet_ntoa
(req_sin.sin_addr));
                    MessageBox(szBuff, "Client Request Data", MB_OK);
                    //发送组播地址和端口给客户端
                    send(wParam, szSend, sizeof(szSend),0);
                }
            }
            else
                if(status==0)
                    MessageBox("Connection was closed by client", "Server",
MB_OK);
        }
        else
        { //网络事件为 FD_CLOSE

            //表示对方已接收到地址信息
            MessageBox("可以发送数据", "success", MB_OK);
            //关闭 socket
            closesocket((SOCKET)wParam);
        }
        return 0;
    }
```

4.3.4.2 数据通道的实现

数据通道用于视频流数据的通信，选用 IP 组播来实现，它主要是基于 UDP 协议的。在此采用海康视频卡附带的 SDK 对网络进行操作。视频数据发送模块用于发送监控视频流。

（1）MP4_ServerWriteData()函数，该函数往发送缓存写数据，网络开发包

通过这个接口获得板卡的数据。为网络数据传输等后续工作做准备。

（2）MP4_ServerSetStart()函数，设置启动捕获的回调，启动网络操作。

（3）MP4_ServerSetStop()函数，设置停止捕获的回调，停止网络操作。

（4）MP4_ServerReadLastMessage()函数，读取客户端 MP4_ClientCommandtoServer 函数发送过来的消息(不超过 900 字节)，消息的内容和长度由用户自己定义。

(5)MP4_ServerStringToClient()函数，给客户端发送消息字符串；返回 TRUE 表示成功，返回 FALSE 表示失败。

（6）MP4_ServerChangeChanType()函数，动态切换通道数据类型；返回 TRUE 表示成功，返回 FALSE 表示失败。

（7）MP4_ClientStart()函数，启动客户端；返回-1 表示失败，其他值表示成功。作为后续操作的参数。

(8)MP4_ClientStop()函数，停止客户端；返回 TRUE 表示成功，返回 FALSE 表示失败。

（9）MP4_ClientGetState()函数，获取客户端状态。

（10）MP4_ClientReadLastMessage()函数，读取服务端 MP4_ServerStringToClient 函数发送过来的消息(不超过 900 字节)。

4.3.4.3 客户端的文件操作

客户端收到服务器端实时传送过来的数据流文件后，除了在本地主机上实时播放该外，还可以保存接收到的视频数据流。下面为相关函数实现，主要采用海康视频压缩采集卡附带的 SDK。

（1）MP4_ClientStartCapture()函数，开始客户端的数据捕获（回调方式，使用 MP4_ClientStart 中的 ReadDataCallBack 函数)。返回 TRUE 表示成功，返回 FALSE 表示失败。

（2）MP4_ClientStartCaptureFile()函数，开始客户端的数据捕获（直接写文件方式）；返回 TRUE 表示成功，返回 FALSE 表示失败。

（3）MP4_ClientStopCapture()函数，停止客户端的数据捕获；返回 TRUE 表示成功，返回 FALSE 表示失败。

4.3.5 视频数据接收播放模块的程序说明及实现

4.3.5.1 多路窗口的实现

视频数据接收播放模块可以通过选择通道接收一路或多路画面，这是一个 MDI 应用。每一路窗口是一个用户界面（UI）线程，负责一路视频流的接收和实时播放。海康视频卡为用户二次开发提供的丰富的 SDK，用以接受服务器端发送的视频数据并实时播放、存储。

图 4.15 视频采集模块功能键

选择菜单命令“操作->接受并播放”，或者直接点击操作面板上的“Start Monitor”键，将会响应 CReceiverDlg::OnOperStart()函数。函数实现中，先调用 MP4_ClientSetShowMode()设定客户端显示模式，如果希望客户端不受显卡的限制，可以把显示模式设置成 OVERLAYMODE 模式，但是这种模式只能开一个监控窗口，或者可以设置为 NORMAL 模式，但对显卡有一定的要求。然后，对结构体 CLIENT_VIDEOINFO 变量 aa0 的各项参数值根据预设赋值，其中包括连接方式、服务器 IP 地址、通道句柄、客户名及密码等等，各项值设定好后，调用 MP4_ClientStart_Card()启动客户端。MP4_ClientRigisterDrawFun()函数用来注册用户视频接收线程，该线程用来管理一路视频流的接受和实时播放。其中 MP4_ServerSetBufNum()和 MP4_ClientSetBufferNum()设置缓冲区大小的，参数改变后，短时间内播放效果可能没有什么差别。缓冲区越大，可能产生的延时越大（一旦出现缓冲区累积的情况），缓冲区太小，在网络不稳定的情况下会出现视频流数据丢失的情况。 MP4_ClientSetPlayDelay()函数的参数改变后，播放效果会有显著的不同。如果 DelayLen 值大于 0，播放器会采用流文件播放方式，播放器尽量保证播放的流畅，可能会延时比较大；如果 DelayLen 值等于 0，播

放器会采用实时流播放方式，播放器尽量减少延时，可能会有不流畅的感觉。客户端必须在开始使用网络开发包时，调用 MP4_ClientSetNetPort()，之后调用 MP4_ClientStartup()；在结束使用网络开发包时，调用 MP4_ClientCleanup()。MP4_ClientSetShowMode()用以设置客户端的显示方式是单窗口还是多窗口，MP4_ClientSetTTL()则用来设置多播的 TTL 参数，这两个函数只能在 MP4_ClientStart() 之前调用。而 MP4_ClientStartCapture() 和 MP4_ClientStartCaptureFile()只能在 MP4_ClientStart()之后调用。

4.3.5.2 通道参数设置

点击“网络->参数设置”，将会弹出通道的参数设置对话框，参数设置窗口必须在接收端处于停止接受的状态下来激活，因为参数的改变必须通过再次调用 MP4_ClientStart()函数来重新连接服务器端才会有效。参数设置窗口如下：

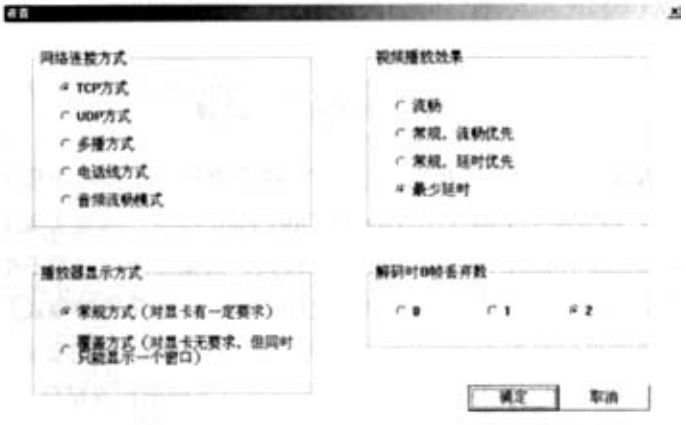


图 4.16 参数设置窗口

函数设计与发送端参数设置流程相仿，重新调用 MP4_ClientStart()函数后，将各变量值传送给相应的 CLIENT_VIDEOINFO 结构变量，各通道将按照新的参数来连接服务器。

4.3.5.3 复位网络通道

当出现缓冲区故障或者网络通道阻塞时，可以点击“网络->清空缓冲区”和“网络->复位网络通道”，调用相关的 SDK 来复位连接，排除故障。相应函数原形如下：

（1）BOOL __stdcall MP4_ClientCleanBuffer(LONG nPort,int nCleanType)，清除数据缓冲区。包括客户端和服务端。

（2）BOOL __stdcall MP4_ClientShut(LPCTSTR m_lAddrIP,char nChannel)，对服务端的 nChannel 通道网络连接初始化，结束当前所有用户对它的访问。

5.3.5.4 视频流解码播放的实现

视频流的解码播放用 DirectShow 实现。前面已经介绍过了，DS-4004HC 视频采集卡将采集到的数据压缩成 H.264 流，通过 IP 组播方式传过来。因此，要建立一个接受并播放 H.264 流的 Filter Graph，如图 4.17 所示。

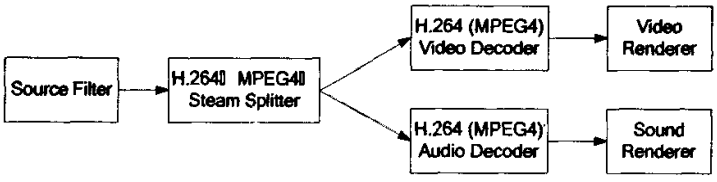


图 4.17 接受播放的 H.264 流的 Filter Graphic

安装海康公司开发的 Hikvision DirectShow Filter，它提供了 H.264 (MPEG-4) 解码的所有 Filter，包括 H.264 Stream Splitter、H.264 Video Decoder 和 H.264 Audio Decoder。这里需要创建一个能接收发送端发来的数据并提供给 FilterGraph 中其它 Filter 解码的 Source Filter。H.264 Stream Splitter 只能工作在拉模式下，因此，Source Filter 也要设计成拉模式。它的设计参考 DirectX SDK 中的 MEMFilte 例子（路径为\DXSDK\Samples\C++\DirectShow\Filters\Async\MEMFile）。海康公司提供的 SDK 封装了对 Filter Graph 的操作，其中创建了两个派生类 CMemStream 和 CMemReader，前者从 CAsyncStream 派生，后者从 CAsyncReader 派生而来。数据的流动主要通过后面的 H.264 Stream Splitter 调用 Source Filter 的输出管脚上 IAsyncReader 接口的对应方法向 Source Filter 请求数据，这些函数代码都无需用户自行编写，以动态连接库的方式提供给用户，方便了用户二次开发。

第 5 章 基于 GPRS 网络的图像传输技术的应用

5.1 无线通讯概述

随着科学技术的飞速发展，各种通信技术也不断发展，由以前的模拟方式发展为更为安全可靠的数字通信方式，由以前的有线通信方式发展为更为快捷方便的无线通信方式。由于无线通信相关技术的保障，正在经历它有史以来发展最快的时期。但是，以往的无线通信技术，传输速率低，可靠性差，远远不能满足视频实时通信的要求。近几年发展起来的第 2.5 代移动通信技术，其传输速率较第 2 代移动通信技术有了很大的提高，在理论上可以实现视频的实时通信。但是在实际应用中，传输速率很多时候依然是瓶颈。在 2.5 代的移动通信技术中，已经在我国推出商用的是中国移动公司的 GPRS 技术和中国联通的 CDMA2000 技术，二者各有所长。由于移动网络的广泛覆盖性和 GPRS 与 GSM 的相关性，我们这里着重向大家介绍 GPRS 技术。

现在全世界使用最广泛的是全球移动系统（GSM）。GSM 是由欧洲主要电信运营者和制造厂家组成的标准化委员会设计出来的，它是在蜂窝系统的基础上发展而成。它的主要业务是语音和数据通信，由于带宽的限制，图像和视频通信的增值业务没有在此系统中大概模商用。

5.2 新一代无线通信网络—GPRS 网络简介

5.2.1 GPRS 简介

包交换网络，特别是因特网（Internet）在 20 世纪 90 年代有了飞速发展。人们设想了带终端和固定计算机在这方面的应用。在 1999 年，通过移动终端接入上述网络有了可能，但仅仅是间接的，因为大部分运行的移动网络是电路式交换，其数据速率受限于 14.4 Kb/s 甚至是 9.6 Kb/s。

在许多的信息应用中，Web 咨询是常用的，一般话务是分散的，一个会议可能持续几十分钟，而数据实际上只是持续几秒，电路模式的数据发送表现出

不方便，在会议期间 BSS 资源被独占，主要是物理性资源。

GSM 主要是电路型的交换，这种交换方式阻碍了网络速率的进一步提高，急需一种新的技术来改变这种状态。

GPRS（General Packet Radio Service）称为无线分组服务，是 GSM 的重要发展。它确定了包交换网络的结构，这个网络具有对移动台的管理功能和无线接入功能，通过包交换和传输速率的提高，GPRS 打开了多媒体移动应用的大门，并使移动通信平滑过渡到第三代。

GPRS 网包括了本身用户、移动用户或固定用户，可以平向建立在不同协议上的各种固定数据网，例如：网络协议 IP（Internet Protocol）、X.25 和 ITU（International Telecommunication Union）的定向联接协议。无论什么样的网络协议都可用术语数据包协议 PDP（Packet Data Protocol）表示。推而广之，网络都被叫做 PDP 网。若干个 GPRS 网也可以互平并提供漫游服务。

在 GPRS 提供 IP 服务时，GPRS 安排一个 IP 地址，IP 的网络域在 GPRS 网上是特定的。可以预见 IP 地址是被动态分配的，就像有时在固定网上的情况一样，网络安排了一个 IP 地址集，这些地址必要时被分配给终端。在 X.25 服务的情况下，终端安排一个 X.25 地址，并可以与另一个 GPRS 网交换同类数据。在所有情况中，一个移动台 GPRS 分配在一个国际移动用户身份 IMSI 上，并被鉴别为带 IMSI 的内部网。

GPRS 建议采用新的基站子系统 BSS 结构，但也确定了一个不同于网络子系统 NSS 的固定网结构，后者更确切地说是在电路交换系统中被提供的。漫游管理重新确定了原则和“GSM-Circuit"结构。有了 GPRS, BSS 就变为多服务接入的子网，它可以联向传统的 NSS 和 GPRS 网，但是开发一个无 NSS 的纯 GPRS 网也是可以的。GPRS 原理取自移动 IP 和蜂窝系统数据包 CDPD（Cellular Digital Packet Data），是在美国开发的。

GPRS 的好处之一就是利用了 BSS 中的统计多路复用，使用了无线频道上的包发送。GPRS 建议还允许使用 TDMA 帧的一个时隙，允许瞬时速率达到近 170 Kb/s。

5.2.2 GPRS 提高速率的技术

GPRS 的优势就是可有高于 GSM-Circuit 的瞬时传输速率。速率的提高取决

于以下几个因素：

•开发了新的移动台设备，它具有在 TDMA 帧多时隙上收发信号的能力量，被称为“多时隙终端”（高速数据交换电路 HSCSD）；

• 缩减对使用者数据的保护；

• 采用了频谱利用率更高的调制方式。

接收中，服务器管理 4 个时隙的终端设备是可以做到的，然而，开发发射中的多时隙终端却有些问题，因为这要增加能耗。这就导致终端自主功能减少和散热问题，然而，GPRS 所针对的应用大多数是服务咨询，信息流量下行方向比上行方向要重要得多，因此对上行频道的限制要大于对下行频道的限制。

如果人们想保持一个可接受的误码率，减少保护和使用虚拟调制方式，则需要一个较高的载干比 C/I。

在一个按某种频率复用制规划的蜂窝网中，移动台受到的干扰略微改变了小区的定位功能。事实上，移动台和干扰基站之间的距离，近似在 D-R 和 D+R 之间变化，即最大相对距离差为 2R/D。然后在一个规则的网络中，如果移动台是靠近第一环上的干扰基站，这就表示它离另一个干扰源较远，干扰的变化相互补偿。反之，由于移动台和基站之间的距离在 D 和 R 之间，则信号 C 有很大范围的变化，这表示载干比有很强的变化，比移动台靠近基站的情况变化更强（不考虑功率控制）。

相应地，一个靠近基站的移动台比蜂窝小区边上的移动台更容易支配高速率。然后，当人们考虑避开现有 GSM 网络结构的限制时，估计 GPRS 可利用的最高速率就可以从 48 Kb/s 升至宣传的 171.2 Kb/s。

5.2.3 GPRS 技术特点

GPRS 服务使人们可以的把 GSM 看作一个数据包传输网，它可以无线接入并有移动终端。GPRS 与 IP 协议和 X.25 协议兼容并存。BSS 结构被重新取定，但支持新功能。专用的路由器 SSGN 和 GGSN 被引入固定 GPRS 网。在无线频道上包发送将节省无线资源，一个终端能被归属于网络，这就是说可以在整个时间内收发数据，无需网络给它分配一个专用无线信道。

GPRS 宣称的最大瞬时速率为 171.2 Kb/s。它对应 8 个时隙，编码形式是 CS4 (171.2 = 8×21.4)。这个速率在实际中是不可能的。

- CS-1 和 CS-2 的编码形式一般是单独使用的；
- 2000 年最优秀的移动台接收中仅管理 4 个时隙；
- 宣称的速率未考虑无用位和起始段。

事实上，物理层允许用 CS-1 传输有用位 181 bit，但高级层（MAC/RLC）只管理字节。有用的块是 176 bit 而不是 181 bit。在每一块上，实体 MAC/RLC 至少加了两字节的起始段。高级层的实体同样也管理着起始段，但其值减少了，因为数据的整体要长得多。人们得到的有用速率见表 5.1。下行频道的最大速率实际是 4×12 = 48 (Kb/s)。

表 5.1 GPRS 信道速率

编码方式	速率（Kb/s）	块的大小（bit）	块的大小（字节）	RLC 数据大小	RLC 层速率（Kb/s）
CS-1	9.05	181	22	<=20	8
CS-2	13.4	268	32	<=30	12

5.3 GPRS 与其他网络的互联

由于无线监控网络中的视频信息很多时候需要接入 Internet，所以需要考察 GPRS 与其他网络的互连。目前 GPRS 与网络连接仍然是采用电路型 GSM 的漫游管理原则。SGSN 和 HLR 交换了 MSC/VLR 和 HLR 交换的相同消息。

5.3.1 GPRS 与各种网络互相联入的基本原理

如图 5.1，讨论了开始断开的一个移动台的情况，它在变换了 SGSN 后联向了新的 GPRS 网。移动台先发出一个申请 ATTACH REQUEST，这是一个属于 GSM 平台上的消息。它的地址定位首先是在 GPRS 网络内部进行更新的，如果有一个电路型的 GSM 和 GPRS 之间的组合路由管理，那么 SGSN 就通知 MSS/VLR，在收到消息 MAP-UPDATE-LOCATION 的基础上要变换定位。举一个 SGSN 和 MSSNLR 之间会话 BSSAP+的例子，即时定位在 MSS/VLR 和 HLR 之间执行，本阶段任务结束后，MSS/VLR 通过消息 BSSAP+通知运行成功的 SGSN。设备 MSS/VLR 和 SGSN 都能分配新的临时身份。新的 TMSI 由 VLR 管理，在消息 BSSAP+ LOCATION UPDATE ACCEPT 中被 MSS/VLR 指明。为了使 MSS/VLR 考虑 TMSI 的释放，移动台发出一个最终释放消息 ATTACH

COMPLETE SGSN 则通过 BSSAP+ TMSI RELOCATION COMPELETE 消息通知 MSS/VLR。C 移动台应该事先脱离一个网络然后接入另外一个网络。例如，移动台可以先脱离电路型 GSM 网络后接入 GPRS 网络。

网络数据包包含内容从 HLR 转移到 SGSN 的用户资料主要有：用户使用的网络协议类型，如 X.25 协议或是 IP 协议等，其所载的 PDP 地址，服务质量水平和 GGSN 地址，而 GGSN 是可以和 PDP 网沟通的。并且一个移动网可以管理好几个协议。在这种情况下，前述信息可以是双工的，联网后，SGSN 认识 GGSN，则每个 PDP 网均可以接入用户，然而 GGSN 不认识移动台。

包含内容的数据报文 GPRS 网络可以给移动台分配一个临时身份发送数据，叫做包临时移动用户身份 P-TMSI，即 Packet-Temporary Mobile Subscriber Identity 这与 TMSI 类似。在一个电路型和包交换型组合的网络中，一个移动台可以被看作两个身份：一个传统的 TMSI 和一个 P-TMSI。另外为了避免混淆，把这两种划分到不同的区域范围内：TMSI 从 0 到 0xBFFF，P-TMSI 从 0xC000 到 0xFFFF。对于 P-TMSI 可以结合签名。这也用数码发送给移动台包括 P-TMSI。对于所有的定位区变化，移动台都在实时的消息中指明签名，且均用均码发送。在签名和 P-TMSI 之间严密性的鉴定保证了安全性，从而免除了用户鉴权。

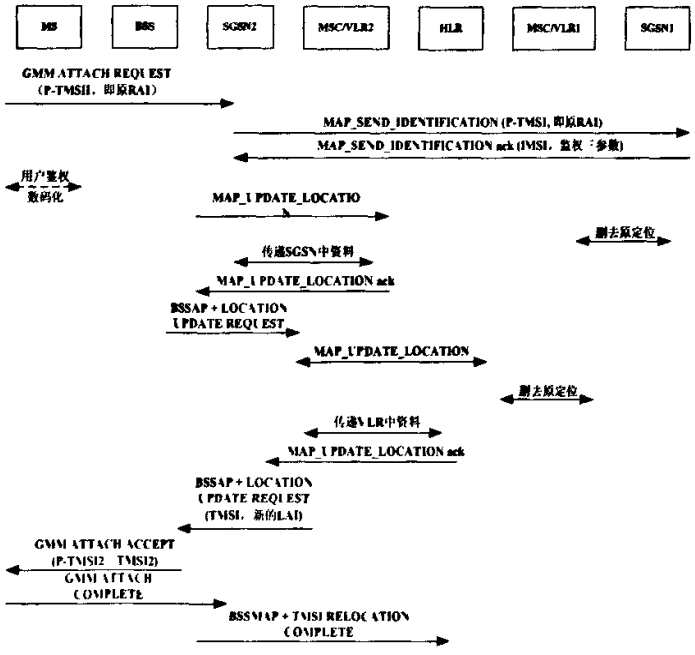


图 5.1 GPRS 与 IMSI 组和联网过程

P-TMSI 是一个短的身份码，它可以在相同的 SGSN 下，清楚的区分两个移动台。然而 P-TMSI 只是在联入网络后才被授予，在连接起始阶段，移动台应该与 SGSN 交换信令，以便与其他的移动台区别开来，但并不发送其 IMSI，为此则采用了临时网络层身份 TLLI，即 Temporary Link Layer Identity。如果一个移动台具备 P-TMSI，则 TLLI 与 P-TMSI 是一致的。只要移动台留在相同的 SGSN 下，它就保持着相同的 TLLI，并被称之为本地 TLLI。如果网络分配一个新的 P-TMSI,则 TLLI 也被更新。

建立网络之间的联入即采用激活式建立一个移动 GPRS 可以支持不同的网络协议，然后，用户可以在同一个终端上开好几个会议，但是服务质量有所不同，人们称有关的数据包协议为 PDP 格式。存储在移动台里的信息集，SGSN，GGSN 为了与 PDP 网络交换数据，一个移动 GPRS 就生成了，必须在 PDP 网络层上激活一个 PDP 格式。这样的话，只可能是移动台联入网络。一般来说，PDP 格式主要包括：所采用 PDP 网络的类型，如 X.25 协议或是 IP 协议等；终端的 PDP 地址；常用用户的 SGSN 的 IP 地址；有用网络服务接入点鉴权 NSAPI，即 Network Service Access Point Identity；协定服务质量。在 PDP 格式激活阶段，终端的 PDP 地址是动态分配的。只存储在 GGSN 中的 SGSN 的 IP 地址，并允许确定数据路由，让来自外部 PDP 网络的数据移至移动台。一个移动台联入 GPRS 由 SGSN 中的 TLLI 鉴别。对于 SGSN 而言，一对 TLLI 和 NSAPI 就确定了一个 PDP 格式，即对一个移动台给出一个格式。实际上 SGSN 有能力从 TLLI 出发重新找到 IMSI。

联网的移动台可以激活 PDP 格式，传送一个消息 ACTIVEATE PDP CONTEXT REQ，该消息包含了除 SGSN 的 IP 地址之外的所有 PDP 格式特性。SGSN 存储着用户特征信息，并恢复 IMSI 和联入已激活 PDP 格式的 GGSN，它接着转移 PDP 格式和 IMSI 至消息 GTP CREAT PDP CONTEXT REQUEST GGSN 存储着 IMSI，PDP 格式和 SGSN 的 IP 地址，移动台在 SGSN 中 GGSN 有使移动台与外部数据网交换数据的能力，并能够通过用户数据通道向 SGSN 发数据。

可能有一个外部 PDP 网络的终端想传输数据给 GPRS 用户，而 GPRS 的 PDP 格式尚未激活。GPRS 向网络提供了激活格式的可能，有必要使 GGSN 记忆用户 IMSI 和 PDP 地址之间的联系。设一个数据包（PDU PDP）达到 GGSN，对它的一个终端而言，格式尚未激活。GGSN 从收件人 PDP 地址开始搜索到用户的

ISMI，然后它向 HLR 申请用户所在的 SGSN 的 IP 地址，当以上这些内容被发送时，GGSN 给 SGSN 发出一个消息，要求移动台激活 PDP 格式。GGSN 可以用缓冲存储器存储 PDP 格式激活结束前达到的数据，然后这些数据发送给移动台。如果移动台尚未联入，HLR 可以给 GGSN 指示，以减少无用信令的传输。

由于 PDP 格式激活过程和漫游区的确定，GGSN 知道哪个 SGSN 与准备状态或等待状态的 GPRS 用户有关，GPRS 的发送机制遵循 IP 移动台协议的原则。那么以下介绍 IP 网用户和 IP 型的 GPRS 用户之间的数据交换，现假设移动台有一个永久性的 IP 地址。

对于 GPRS 网络与其他公用网络之间的互联其大致过程就是 IP 数据网发送数据包到网络，其中数据包内包括了其 IP 地址以及相关的数据。在 IP 网络中，GGSN 根据数据包中的 IP 地址选择合适的路由，建立通道，并对传输给的数据包加载更多的传输信息。加上有关的 SGSN 地址、起始段、IMSI、NSAPI 在数据包的报文头。通过 SGSN 地址找到与移动台相关的 SGSN，并建立通道节点之间的连接，按照提供的 IP 地址发送给正确的移动用户并建立完整的连接通道从而与用户建立网际之间的连接。同样的，由用户提出建立与各种网络之间的连接的过程则与前面提到的相反，建立是由用户主动发出，经由 GGSN 重新加载数据包文的信息找到 IP 地址所对应的网络用户并建立连接从而构成完整的网络连接。

5.3.2 上行联入即网络呼叫移动用户建立网络连接

当一个数据网的固定终端把 IP 数据包发向一个移动用户时，该用户即被引向注册 GPRS 网的 GGSN 上。如果对于该用户没有任何一个格式被激活，则 GGSN 执行网络对 PDP 格式的激活的过程。反之，GGSN 找到用户所在的 SGSN 的 IP 地址，用户的 IMSI 和已被激活的 NSAPI，构成了带 IMSI 的起始段，NSAPI 和数码化变量的消息。起始段被连向数据包 IP，以便形成一个 PDU GTP。允许用 IMSI 和 NSAPI 以统一方式鉴别 PDP 格式，对 GGSN 和 SGSN 而言也同类。PDU GTP 或是通过 TCP，或是通过 UDP 被传送到 SGSN PDU GTP 或 UDP 被打包在 IP 数据包中，这个数据包中有作为收件人地址的 SGSN 的 IP 地址，GPRS 移动台位于其中。当接收到数据包后，SGSN 就解包并由 IMSI 和 NSAPI 鉴别收件移动台的 PDP 格式，它就在数据链路上传输数据包给移动台。SNDCP 实体可

能执行一个压缩而 LLC 实体进行数码化，根据移动台 GMM 的状态，网络在各漫游区上呼叫移动台。在其它网中漫游时其原理是相同的，涉及到的 GGSN 总是用户注册网络的 GGSN.数据包总是从注册网络的 GGSN 流向被访问网的 SGSN，而访问是通过骨干网的相关媒体进行的。

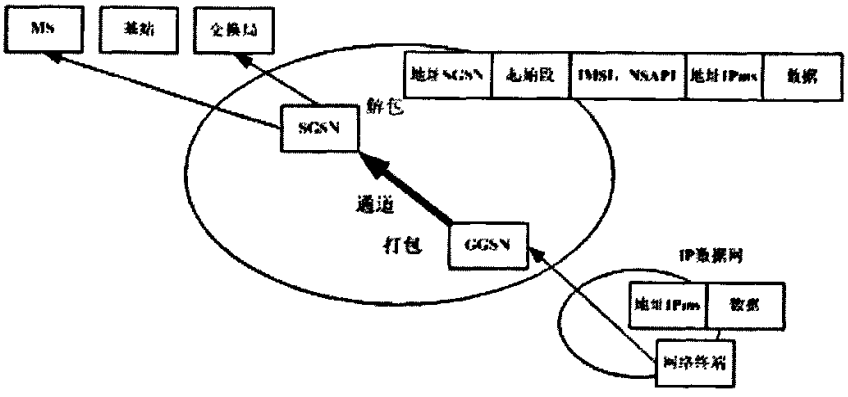


图 5.2 网络与移动终端的网络联入以及数据传输

5.3.3 下行联入即用户呼叫建立网络联入

当移动台发送一个 IP 数据包或是更普遍的发送一个 PDP 包时，数据包被 NSAPI 和给定的 TLLI 发向 SGSN, SGSN 就重新找到用户格式并推演出所涉及的 GGSN 地址，包被传输到 G-PDU，其起始段包含了 IMSI 和 NSAPI，可由此鉴别格式。而数据包则通过从 SGSN 到 GGSN 的通道，GGSN 再把数据包传给收件人。

5.4 视频流的无线传输方案分析

本章节的研究内容主要是远程无线视频传输，是上一章视频图像有线传输的扩展应用研究。采用客户加服务器模式，与有线传输模式类似，该系统的远端是接入 Internet 的计算机作为客户端，本地端则是插有海康视频卡的 PC 机作为服务器，考虑到 Windows 操作系统的使用较普遍、功能完善、编程资料多等特点，本系统决定两端都采用 Windows XP 操作系统。

要特别说明的是，客户端计算机必须连接在以太网上，而且 IP 地址固定。本地服务器端则通过 RS232 串口与 GPRS 模块连接，并且 GPRS 模块的 SIM 卡

也必须绑定固定的 IP 地址，这样有利于简化应用程序编写，以免因本地服务器端每次开机都出现 IP 地址的变化而导致客户端在每次登录前都要重新确定服务器端的 IP 地址。

5.4.1 视频无线传输模块的总体设计及实现

视频无线发送端工作流程如图 5.3 所示。考虑到利用 GPRS 无线传输数据的高额成本，系统的服务器端是被动地等待客户端登陆，一旦登陆成功，发送端开始通过 RS232 串口向 GPRS 模块发送数据，而接收端就打开接收视频信号的线程等待视频信号的到来，解码并播放服务器端发来的视频信号。

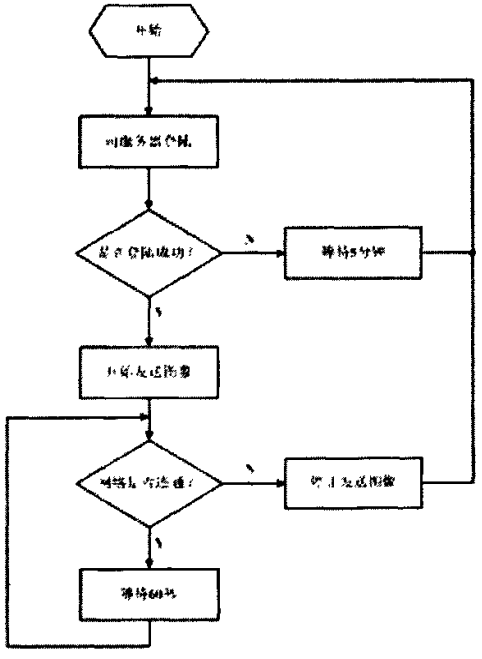


图 5.3 无线模块服务器端的主线程工作流程

在本方案的实现中，视频的编码是通过视频压缩卡——硬件编码的方法来实现的。采用硬件编码的方法主要是为了提高数字视频信号采集、编码和记录的速度以及在网络中传输的实时性和传输的服务质量(Q_oS)。如果采用软件的方法来实现视频的编码，势必会增加系统 CPU 资源和内存资源的开销。当多路视频同时采集、编码，并且将视频数据记录到硬盘，或通过网络传输，或者两个

任务同时进行，这时系统将不堪负荷。轻者将造成视频传输延迟、视频回放抖动，重者将影响系统的稳定性，直至系统崩溃。

采用硬件压缩的方法来进行应用软件的开发时，硬件厂商提供了 SDK 软件包。该 SDK 软件包为我们提供了相应的输出函数用于从视频压缩卡中获取视频数据流，并且数据包的大小也可预先设置。该视频数据包的大小会影响到视频在网络中传输的实时性和视频在接收端回放时抖动的程度，因此该视频数据包大小的设置应该是传输时的实时性和回放时的抖动情况的折中。

由于 UDP 协议是不能保证传输时视频数据包的有序和无重复，因此，必须将取得的视频流采用 RTP 协议封装以后再在网络中传输，以保证视频数据报文的有序和无重复。再将采用 RTP 协议封装好的视频流移至发送缓冲区进行传输。

由于以上从“压缩编码”一直到“视频传输”的整个过程都是可逆的，因此，在视频接收端进行逆向操作，可以在客户端实现视频图像的重建。如图 5.4 所示是本节中采用的无线网络视频传输的基本框图。

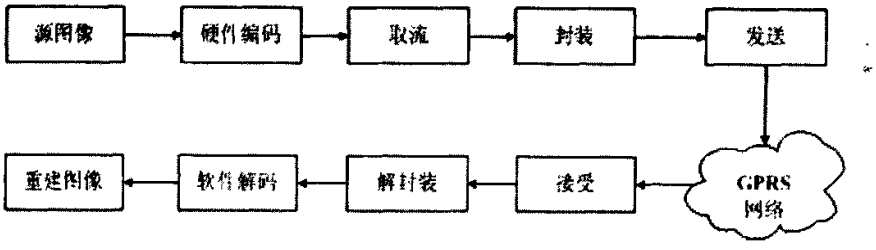


图 5.4 无线网络视频传输的基本框图

5.4.2 视频无线传输模块的硬件电路

远程终端利用了现有的 GPRS 网络来实现图像和数据的无线传输，具有方便、简单、可靠等优点。因此在设计系统时候，选择一种合适的 GPRS 模块也就成了关键问题之一。在本子系统中，GPRS 模块主要完成与 PC 机之间的数据交换以及移动公司的 GPRS 网络与远端计算机之间的数据交换。前者直接通过 RS232 串口按照系统协议传输数据，后者可以通过 TCP、UDP 协议实现。因此要选择的 GPRS 模块应该支持 TCP、UDP 协议。现在市场上 GPRS 模块的生产商很多，如西门子，索尼-爱立信，摩托罗拉等等。经过比较和研究，采用西门子公司的 MC35 模块，它使用方便，有很好的技术支持，性价比高，且模块支

持多种传输协议。

对 MC35 添加必要的硬件外围电路，MC35 通过 RS232 串行接口直接与服务器端 PC 机相连，PC 机通过 AT 指令控制 GPRS 模块的工作模式，视频数据也是通过串口发送给 MC35 模块的。

5.4.2.1 GPRS 通讯处理模块

目前市场上关于 GPRS 的应用主要使用的是西门子的 MC35 模块，该模块结合语音、数据传输、简讯服务及 FAX 等功能，最大传输速率可达 85.6Kbps，在通讯状况良好的情况下，完全能满足监控视频传输的需要。模块集成天线、RF、Base band、快闪内存等组件，并以 40 个 pin 脚外接，支持 RS232 等。根据系统要求及性能价格比，在本系统中 GPRS 模块选用 MC35。

MC35 有丰富的 AT 命令，功能强大，操作灵活方便。是继 GPRS 手机外又一种重要的 GPRS 移动通讯系统的终端设备。它是传统调制解调器与 GPRS 无线移动通讯系统相结合的一种数据终端设备，因此，也叫无线调制解调器。它的出现给 GPRS 的发展注入了新的活力。该模块集射频电路和基带于一体，向用户提供标准的 AT 命令接口，为传输数据、语音、短消息和传真提供快速、可靠、安全的传输，方便用户的应用开发和设计。

（1）它的各项指标如下：

① 双频带：EGSM900/GSM1800

② B 类 GPRS 移动台

③ 设计简小、易于集成

④ 适用于 GSM 2/2+

⑤ 输出功率：功率级 4（2W）在 EGSM900/功率级 1（1W）在 GSM1800

⑥ AT 命令控制

⑦ 波特率：可选波特率 300bps~115kbps，自动波特率 4.8~115kbps

⑧ SIM 应用工具箱

⑨ 工作电压范围：3.3~5.5V

⑩ 功耗：空闲模式（EGSM900/1800）：10mA/10mA

-语音模式（EGSM900/1800）：300mA（均值）/2.0A（峰值）

-睡眠模式：3mA

-掉电模式：100

-重量：18g

-体积：54.5×36×6.76mm

-正常工作温度范围：-20 ℃ ~ +55 ℃

-存放温度：-20 ℃ ~ +85 ℃

西门子的 MC35 的外观如图 5.5 所示。

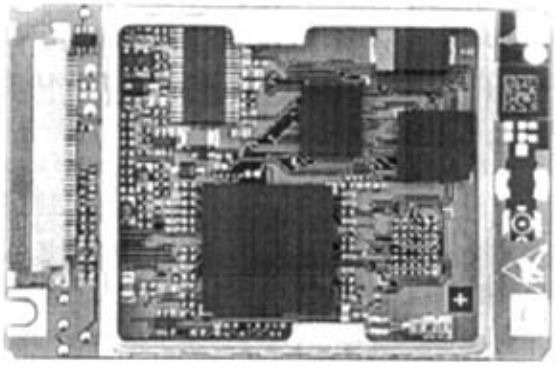


图 5.5 西门子 MC35 GPRS 模块外观图

（2）可提供的数据传输形式：

① SMS（短消息）：点对点 MO 和 MT；SMS 单元广播；文本和分组数据单元模式；

② Fax（传真）：组 3.2 类；

③ Data（数据）：CSD 传输速度 2.4，4.8，9.6，14.4kbps；USSR；非透明模式；V.110；GPRS 最大数据传输速率：85.6kbps（下行链路）；编码方案：CS 1，2，3，4；PPP 协议栈

（3）MC35 的数据接口：MC35 的数据输入/输出接口实际上是一个串行异步收发器，它符合 I TU-T RS232 接口标准，它有固定的参数：8 位数据位和 1 位停止位，无校验位，波特率在 300bps~115kbps 之间可选，硬件握手信号用 RTSO/CTSO，软件流量控制用 XON/XOFF，CMOS 电平，支持标准的 AT 命令集。通过这一接口可以用 AT 命令切换操作模式，可以使它处于语音、数据、短信息或传真模式。

（4）引脚功能：MC35 共有 40 个引脚，通过一个 ZIF（Zero Insertion Force）连接器引出。这 40 个引脚可以划分为 5 类，即电源、数据输入/输出、SIM 卡、

音频接口和控制。第 1~14 脚为电源部分，其中 1~5 脚为电源电压输入端 Vbatt+，6~10 脚为电源 GND，11、12 脚充电引脚，13 对外输出电压（共外电路使用），14 为 ACCG_TEMP 接负温度系数的热敏电阻；24~29 为 SIM 卡引脚，分别为 COIN，CCRST，CCIO，CCCLK，CCVCC 和 CCGND；33~40 为语音接口用来接电话手柄；15，30，31 和 32 脚为控制部分，15 为点火线 IGT（Ignition），当 MC35 通电后必须给 IGT 一个大于 100ms 低电平，模块才能启动，30 为 RTC backup，31 为 Powerdown，32 为 SYNC，16~23 为数据输入/输出，分别为 DSRO，RINGO，RxDO、TxDO、CTSO、RTSO、DTRO 和 DCDO。SIM 卡引脚个定义如图 5.6 所示。

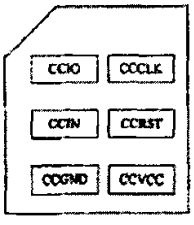


图 5.6 SIM 卡的引脚图

对于 GPRS 移动台必须有一个用户识别模块（SIM 卡），SIM 卡上包含了所有用户的信息。个人识别码（PIN）可以防止 SIM 卡未经授权而使用，每当移动用户开机时，GPRS 或 GSM 系统先要自动鉴别用户的合法性，只有在系统认可以后，才为该移动用户提供服务。MC35 使用外接式 SIM 卡，ZIF 连接器上有 6 个引脚作为 SIM 卡的接口，SIM 卡上也有 6 个引脚与他们相对应。

（5）SIM 卡同 MC35 的连接方法：SIM 卡上的 CCRST、CCIO、CCCLK、CCVCC 和 CCGND 通过 SIM 卡阅读器与 MC35 的同名端直接连接，COIN 悬空，而 MC35 上的 COIN 通过一个 3.3 KΩ电阻与 CCVCC 相连。这种连接方式是由 SIM 卡阅读器决定的。SIM 卡安装好以后，再到中国移动申请开通 GPRS 数据业务，就可以通过 GPRS 进行数据传输了。

（6）MC35 的内部结构：MC55 内部的主要模块包括 GSM 基带处理部分，GSM 射频部分，电源 ASIC 部分，Flash ROM，SRAM，ZIF 接口和天线模块。MC35 的内部框图如图 5.7 所示。

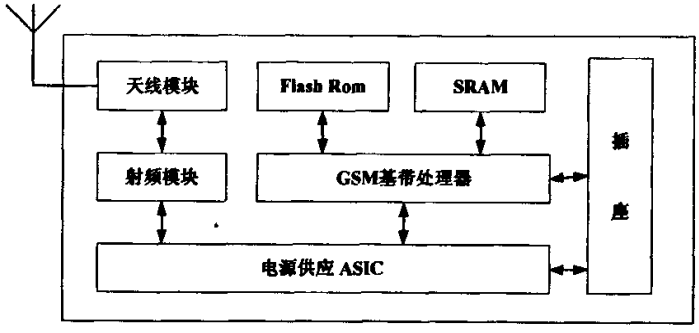


图 5.7 MC55 内部模块结构框图

模块的基带处理器部分处理模块内部的语音、信令和数据传输，控制内部软件运行接口以及通过一个异步串口和主机通信。基带处理器包括一个芯片再加上信号处理 IC，主要处理蜂窝电话的模拟和数字信号。为了满足移动用户日益增长的需要，MC35 模块支持 FR、HR 和 EFR 语音和信道编码而不需要改动外部硬件。它的高级程度降低了系统复杂度、缩小了板的尺寸以及减小了元件数量。基带信号处理器主要集成了一下部件：

① C116 微处理器内核；

② 数字信号处理器（DSP）内核；

③ 在片微处理器 SRAM 可灵活的配置为程序存储器或数据存储器；

④ 可编程 PLL；

⑤ GMSK 模块、自动频率控制（AFC）；

⑥ 硬件 Viterbi 加速器；

⑦ 带 A/D/A 的音频和带通滤波器；

⑧ 串行射频控制接口。

5.4.2.3 硬件部分系统框图

如图示，视频采集的服务器端通过 RS232 串口直接与 GPRS 模块 MC35 连接，通过 AT 指令集传输控制指令和数据。SIM 卡上的 CCRST、CCIO、CCCLK、CCVCC 和 CCGND 通过 SIM 卡阅读器与 MC35 的同名端直接连接，COIN 悬空，而 MC35 上的 COIN 通过一个 3.3 KΩ 电阻与 CCVCC 相连。

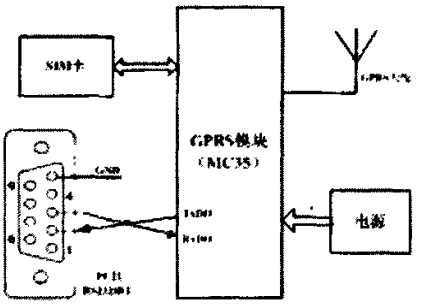


图 5.8 无线图像传输部分硬件框图

5.4.3 视频无线传输模块的软件结构

在本课题中，视频压缩卡的厂商给应用软件开发者提供了系统 SDK，是专门为该系列一路及多路板卡设计的本地录像软件接口程序，以动态库的形式(*.dll)提供。其中包括板卡的初始化、设置流类型、获取视音频流等一系列相关函数，用户可直接调用。

无线视频传输的主要任务是实现取流、封装到发送以及接受、解封装、软件解码和重建图象的过程。采用基于单线程的编程机制可以实现这一过程，但是效率低，不能保证网络视频传输的实时性和视频传输的质量。因为基于单线程的编程实现，发送方必须先将视频信号进行压缩编码，然后将压缩编码以后的数字视频封装成 RTP 视频数据包，最后才能移至发送缓冲区进行发送，如此周而复始，发送一个又一个视频数据包。很明显，前一传输过程不能和后一取流封装过程同时进行，这就造成了 CPU 的并行效率低下，从而降低了数字视频在网络中传输的整体效率。

32 位操作系统抢先式多线程任务机制的推出解决了 CPU 并行效率低下等问题，使得数字视频在网络中的实时传输实现成为可能。它利用线程这个最小执行体作为 CPU 分配时间的实体，可以同时运行多个线程，并按线程的优先级高低调度多个线程。这就使得可以在本文中采用多个线程同时实现取流、封装、传输等多个过程，避免了一时只能实现单一过程的不足，从而大大提高了 CPU 的并行效率，同时也提高了视频图像传输的效率和质量。

由于视频图像传输需要做到强实时性和高传输质量，因此，在视频图像的

发送端，本文采用了基于事件驱动的三缓冲区多线程结构，即采用取流缓冲区、封装缓冲区和发送缓冲区等三个缓冲区，分配了取流封装线程、内存切换线程、视频图像发送线程和程序主线程等四个线程，利用了取流缓冲区空、取流缓冲区满、封装缓冲区空、封装缓冲区满、发送缓冲区空、发送缓冲区满及允许发送等七个事件。如图 5.9 所示是基于事件驱动的三缓冲区多线程结构的基本模型。

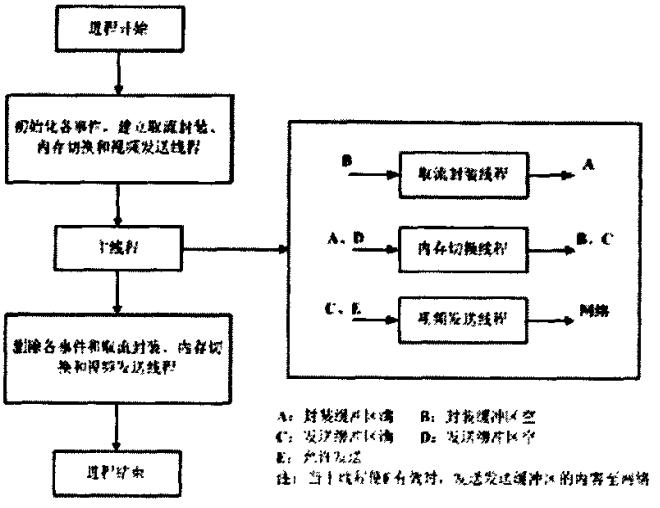


图 5.9 基于事件驱动的三缓冲区多线程结构模型

在基于事件驱动的三缓冲区多线程结构模型中，取流缓冲区用于存放硬件完成压缩编码的码流，封装缓冲区用于存放完成了 RTP 协议封装的码流，发送缓冲区用于存放待发送的数据；取流封装线程负责从视频压缩卡中取出视音频码流，并用 RTP 实时传输协议进行封装；内存切换线程负责将封装缓冲区的数据切换到发送缓冲区；视频图像发送线程负责将发送缓冲区的内容发送至接收端；主线程负责系统的总体调度并协调上述三个线程的运行。线程与事件之间的相互关系是:

（1）当封装缓冲区空事件有效时，激活取流封装线程，待封装结束后，使封装缓冲区空事件无效，封装缓冲区满事件有效；

（2）当封装缓冲区满事件且发送缓冲区空事件有效时，激活内存切换线程。然后使封装缓冲区满事件和发送缓冲区空事件无效，封装缓冲区空事件和发送缓冲区满有效；

（3）当允许发送事件且发送缓冲区满有效时，激活视频发送线程发送缓冲区的数据，然后使允许发送事件和发送缓冲区满事件无效，并使发送缓冲区空事件有效，同时将视频数据报文发送至网络；

（4）主线程在接收到系统的有关事件或消息时，使允许发送事件有效，并协调上述三个线程的运行和处理来自用户的其它信息。

在视频数据接收端应用程序中，有着基本相似的多线程多缓冲区事件驱动结构，所不同的是：发送端采用硬件(图象采集卡)对源视频编码，接收端则采用软件解码。

5.5 小结

第 4 章、第 5 章主要以“基于 H.264 的视频监控系统”为实例进行“图像传输技术的研究”，就传输方案来讲，主要分为两个部分——有线网络传输和无线网络传输。

有线网络图像传输，主要要解决图像压缩编码和网络传输速度的问题。课题中采用了海康公司的 DS-4004HC 视频压缩卡，该视频卡不但能实时提供压缩比极高的 H.264 视频流，还提供了大量的 SDK 供用户二次开发，从而解决了图像压缩的问题。要将图像视频在 Internet 上实时传输，足够的带宽是必须的，现今的网络远程传输主要通过局域网、广域网、互联网，网络的普及程度和网络传输速度基本上能满足视频传输的需要。经过封装后的视频信号在 IP 网络上的传输，实时性要求高，而可靠性要求稍低一些，因此本课题采用 TCP 与 UDP 相结合的方法来实现图像传输。对于可靠性要求较高的控制数据，采用 TCP 协议通讯，而对于数据量大，可靠性要求较低的视频数据流，则采用 UDP 协议传输。

无线网络图像传输，主要借用无线公网，将服务器通过 GPRS 网络接入 Internet。该方案具有很多优势，具有结构简单、使用便利、移动性强、适用场合多等特点，有非常广阔的发展前景。与有线传输方案比较，服务器端增添了 GPRS 模块及相应的外围电路，服务器将视频压缩卡采集来的视频流按照相应的网络传输格式封装起来，通过 RS232 串口发送给 GPRS 模块。由于 GPRS 网络接入了 Internet，对每个模块，网络都会分配它一个 IP 地址，所以，对于客户端（接收端）来讲，有线与无线图像接收过程都是相同的，客户端只需要通过 IP 地址，加入服务器的网络视频组播，即可接收视频图像。

第6章 总结与展望

图像传输技术的是一个涉及内容广泛的研究领域，并有着非常广阔的应用前景。本文在阐明图像传输过程中所用到的关键技术和相关理论后，以“基于H.264的视频监控系统”为实例，给出了图像传输的具体应用方案。

现今，市场上关于图像传输系统的方案和成品虽然不少，但从考虑成本和实用性的角度出发，很多方案的广泛应用受到了一定程度的限制。譬如，虽然纯软件的图像传输系统成本投入少，但由于实时性方面的欠缺，目前只在视频娱乐方面有所应用。而基于嵌入式的图像传输系统，在实时性、实用性方面有所提高，可高额的成本投入却令人望而却步。本文给出的视频监控系统实例，主要讨论了两种图像传输方案，各有特点，用户可根据自身的需要选择应用。

本课题研究的是如何实现基于H.264图像传输系统的一些相关的关键技术。如实现了基于H.264协议的压缩编/解码器，应用了流媒体技术以及相关网络协议并以实现。在本论文中除了详细论述了所应用的技术之外，还着重讨论了系统各个组成部分的实现、系统中主要组成部分之一的视频服务器的研究，并提出了TCP与UDP相结合的H.264视频流的传输方法，以及基于GPRS网络的无线图像传输方案的应用研究。

但由于本系统涉及到的相关技术比较多，如H.264视频压缩编/解码、流媒体技术、多媒体通信技术、网络技术、嵌入式应用技术等，因而涉及到的程序代码也比较复杂。系统实现后可以感觉到存在着不足点，对本系统的一些改进和优化是以后要做的工作。

下一步，本课题要完成的任务有：

（1）视频编解码技术仍然是制约图像传输技术应用的重要因素，压缩比不高导致大量的通讯带宽用于视频传输。编解码速度过慢影响了图像传输的实时性。随着最新的数字编码技术的不断成熟和完善，多级图像传输网络将得到更广泛的推广和应用。

（2）网络图像传输对网络本身提出很高的要求。即使有了强大的图像压缩设备，没有足够带宽的网络，系统建设仍然遥不可及。好在网络技术发展已经大大超前，千兆光纤以太网、百兆到桌面已经成为现实。数百路图像的传输可

以得到有效保证。因此，在未来的需要视频监控的场所建设初期，必须积极考虑千兆光纤网的建设，并把主要用途放在图像传输方面。

（3）继续跟踪移动通信的发展趋势，对 3G 网络作进一步的研究，探索在 3G 网络中进行多媒体通信的方法。进一步完善系统的整体功能，增强各软件模块的独立性。研究多媒体通信的 QoS 控制技术。

（4）研究基于无线网络和嵌入式系统的视频传输技术。随着数字信号处理技术和 DSP 芯片的发展和应用，现在基于 PC 结构的监控系统的部分功能将由嵌入式系统完成，摆脱 PC 机和传输线缆的束缚。快速的硬件计算能使图像传输系统的时延减少、图像的清晰度更高，设备体积更小、能耗更低、更方便携带，真正做到“一机在手中，通晓天下事”。

总而言之，在这个项目的研究和开发过程中，作者深深体会到构建一个图像传输系统是很复杂和庞大的工程，因为它涉及多方面理论和技术，而其良好的应用前景，又使越来越多的人投入到相关的研究开发中。

致谢

在论文完成之际，首先向我的导师钱雪军副教授表示衷心的感谢！

在同济大学攻读硕士学位的两年半时间中，钱老师广博的学识、严谨的治学态度、不懈努力、诲人不倦、锐意进取的精神品质，都令人难以忘怀。无论在学习上还是在生活上，钱老师都给了我无微不至的关怀和帮助。他在学习工作中帮我分析问题，给我不断的鼓励和开拓的信心；生活中平易近人，关心本人的生活和学习，营造了良好的工作和学习氛围。这一切都值得我在今后的工作和生活中学习，是我宝贵的财富。我在攻读硕士学位阶段所取得的成绩都凝聚着钱老师辛勤的汗水。在此，借这篇论文对钱老师表示深深的敬意和由衷的感激！

同时也要感谢郎诚廉副教授，谢谢他对我的学习工作给予的耐心细致的指导。从与他的交往中我获益良多。

感谢本实验室的王朝晖、臧雨霖、毕莹玉、蒋大炜、黄晓明、曹季烽、周丹丹等师兄弟姐妹。感谢他们对我生活和学习上的帮助。

在此我要特别感谢张嘉、韦华、刘海龙和陈熙熙同学，许多次是他们让我从失败中重拾信心，坚持至今。

最后我要感谢我的父母和所有的亲人。他们为我的成长付出了无数的心血。他们的殷切期盼给了我奋发向上的动力，他们的亲情赋予了我战胜挫折的勇气。

谢谢各位！

杨文斌

2007 年 3 月

参考文献

[1] 刘峰. 视频图像编码技术及国际标准. 北京邮电大学出版社，2005

[2] e 通科技研究中心. 网络视频视频技术及应用标准. 北京：人民邮电出版社，2005

[3] 刘富强. 数字视频信息处理与传输教程. 北京：机械工业出版社，2001

[4] 苏广大. 数字图像处理系统. 北京：清华大学出版社，2002

[5] 毕厚杰，汪涛宽带. IP（数据）和视频接入技术. 北京邮电大学出版社，2002

[6] 张江山，鲁平. 视频会议系统及其应用. 北京：北京邮电大学出版社，2002

[7] 朱洪波 等. 通用分组无线业务（GPRS）技术及应用. 北京：人民邮电出版社，2005

[8] 文志成. GPRS 网络技术. 北京：电子工业出版社，2005

[9] 窦中兆，雷湘. CDMA 无线通讯原理. 北京：清华大学出版社，2003

[10] 王超龙，陈志华. Visual C++ 6.0 入门与提高. 北京：人民邮电出版社，2002

[11] 杨红云，伊立民 等. Visual C++程序设计视频教程. 北京：电子工业出版社，2005

[12] 刘祎炜. Visual C++ 视频/音频开发实用工程案例精选. 北京：人民邮电出版社，2004

[13] 李峰，杨磊. 闭路电视监控系统. 北京：机械工业出版社，2002

[14] 胡栋，朱秀昌，刘峰. 数字图像处理与图像通讯. 北京：北京邮电大学出版社，2002

[15] 王汇源. 数字图像通信原理与技术. 北京：国防工业出版社，2000

[16] 张春林. 监控系统中数字视频技术的研究. 上海交通大学出版社，2000

[17] 邵贝贝. 单片机技术的发展与单片机应用的广泛选择. 北京：电子技术应用 1993

[18] 徐安，郭其一，岳继光等. 微型计算机控制技术. 北京：科学出版社，2004

[19] 何立民. 建设单片机应用平台，实施平台发展战略. 北京：今日电子，2000(2)

[20] 冯继超. 面向 21 世纪的嵌入式系统及发展方向. 南京：工业控制计算机，2001(5)

[21] 周航慈.《单片机与嵌入式系统程序设计技术》演讲稿，2003

[22] 徐安. 单片机原理与应用. 北京希望电子出版社，2003

[23] A.Murat Tekalp, A.Marat. 数字视频处理. 北京：电子工业出版社，1998

[24] 余兆明，查日勇，黄磊，周海骄. 图像编码标准 H.264 技术. 北京：人民邮电出版社，2006

[25]（美）IAIN E.G.RICHARDSON. H.264 和 MPEG-4 视频压缩：新一代多媒体的视频编码技术. 国防科技大学出版社，2004

[26] INTERNATIONAL TELECOMMUNICATION UNION. TELECOMMUNICATION STANDARDIZATION SECTOR OF ITU —— H.264. 2003

[27] Mika Helsingius. Image comp ression using multiple transforms. Signal Pr-ocessing:Image communication, 2000

[28] Dengsheng Zhang ,Guojun lu. Shape-based image retrieval using generic fo-urier descriptor. Signal Processing:Image communication,2002

[29] Changick Kim. Content-based image copy detection Signal Processing:Image

communication, 2003

[30] Mitchell,Joanl. MPEGVideo:compressionstandard. NewYork:Chapman&Hall,1999

[31] G.Franceschini,The Delivery Layer in MPEG-4, Signal Processing:Image communication,2000

[32] C.Herpel,A.Eleftheriadis. MPEG-4 Systems:Elementary steam management Sig-nal Processing:Image communication,2000

个人简历　在读期间发表的学术论文与研究成果

个人简历：

杨文斌，男，陕西人，1982 年 6 月生。

2004 年 7 月毕业于 同济大学　电气工程及其自动化 专业 获学士学位。

2004 年 9 月进入　同济大学　电力系统及其自动化 专业 读硕士研究生，攻读硕士学位。

已发表论文：

[1] 杨文斌，钱雪军. 基于 GPS、GPRS 的铁路道口预警装置的研究. 铁路计算机应用，2006