专 业 学 位 硕 士 学 位 论 文

基于 FFmpeg 和 SDL 的智能录屏及播放系统

Intelligent System of Screen Recording and Playing Based on FFmpeg and SDL

作 者 姓 名：席铮

工 程 领 域：电子与通信工程

学　　号：31609042

指 导 教 师：王洪玉 教授

完 成 日 期：2018.10

大连理工大学

Dalian University of Technology

摘　要

近三年来，随着网络和通信技术的飞速发展、硬件设备性能的不断提升，使用视频流媒体技术结合各种智能硬件使得直播、无线同屏等功能可以更普及、更方便、更智能。同时，随着个人及企业对直播、安防性能上的要求不断提升，研发低延时、无卡顿、高清晰且可跨平台、多终端可使用的智能录屏及流媒体播放系统具有极大的理论研究意义和应用价值。

本文在对流媒体技术和智能硬件研究的基础之上，根据实际直播和监控安全需求，提出了一种基于FFmpeg和SDL的智能录屏及播放系统。整个系统采用模块化设计方法，分为视频流推送端、服务器端和视频流接收端，各模块之间使用API进行链接并协同工作，具有传输稳定、便携、扩展性强以及并发性强的优点，可在大型教室、大型会议室、无外网环境等诸多场景下应用。本文所涉及到的研究工作可分为以下几个部分：首先提出一种基于Screen Capture Recorder和DirectShow技术的屏幕捕捉方式，可以处理多种信息源、不同格式以及适配多种硬件设备，有效提升了原始视频流的质量；其次，利用FFmpeg完成视频流的推送和接收。采用H.264视频编解码技术和GPU硬件加速技术，保证了视频流的平滑性，提高了系统的流畅性。最后，通过调用Filter模组以及利用反馈机制使得接收视频流端支持不同平台下的不同特性，如屏幕大小适配、分辨率设置等。

为了验证本文提出的基于FFmpeg和SDL的智能录屏及播放系统的性能，在开放环境、大教室、小会议室等环境中对系统的基本功能进行了相关实验，同时通过对各项关键指标的测量，如传输距离、传输延时、视频抖动程度，对系统进行综合性的评估。结果表明，本系统在一般环境和极端环境下均可稳定运行，传输距离及延时可满足需求，是可行的录屏及流媒体播放案例。

关键词：FFmpeg；SDL；智能录屏；实时播放；流媒体

Intelligent System of Screen Recording and Playing Based on FFmpeg and SDL

Abstract

In the past three years, with the rapid development of network and communication technology and the continuous improvement of hardware equipment performance,the use of video streaming media technology combined with various intelligent hardware makes the functions of live broadcast and wireless co-screen more popular, convenient and intelligent.In the daily entertainment and security areas, there are broad prospects for development.

Based on the research of convection media technology and intelligent hardware, this paper proposes an intelligent system of screen recording and playing based on FFmpeg and SDL according to the actual live broadcast and monitoring security requirements. This system USES the method of modular design, the system can be divided into push streaming video terminal, server, and streaming video, use the API link between each module and collaborative work, stable transmission, portable, the advantages of strong extensibility and concurrency is strong, can be in the large classroom, large conference room, without the network environment, and many other application scenarios.The main research work of this paper is divided into the following parts: first, this paper proposes a Screen Capture method based on Screen Capture Recorder and DirectShow technology, which can process multiple information sources, different formats and multiple hardware devices, effectively improving the quality of original video streaming. Secondly, FFmpeg is used in this system to push and receive video streams. Among them, h.264 video encoding and decoding technology and GPU hardware acceleration are used to ensure the smoothness of the code stream and improve the smoothness of the system. Finally, by calling the Filter module and using the feedback mechanism, the receiving video stream terminal supports different features under different platforms, such as screen size adaptation and resolution Settings.

In order to validate the presented intelligent system of screen recording and playing based on FFmpeg and SDL , respectively, in the open environment, such as large classrooms, small meeting room environment basic function of this system has carried on the field test, at the same time through a survey of the key indicators, such as transmission distance, transmission delay, degree of video jitter, for comprehensive assessment of the system. The results show that the system proposed in this paper can run stably in general environment and

extreme environment, and the transmission distance and delay can meet the demand, which is a feasible case of screen recording and streaming media playback.

Key Words：FFmpeg; SDL; Intelligent screen recording; Real-time playback; Streaming media

目　　录

1 绪论

1.1 课题背景及意义

近年来，随着网络带宽的不断增加和视频压缩技术的飞速发展，视频流媒体技术在互联网上越来越受欢迎，并在逐渐取代传统的信息媒介[1]。网络上的传统信息，例如：静态图片和文字已经不能满足中国网民对信息的日常需求。因此，视频流媒体技术应运而生。流媒体的出现极大地便利了人们的工作和生活，并且可以在观看视频的同时下载它。

流媒体是指利用流式传输技术，在网络上连续传输的媒体文件。流式传输技术是将视频等多媒体文件编码并压缩成压缩包，由流式服务器连续传输到计算机[2]。与传统的多媒体应用相比，流媒体的应用具有以下特点：

（1）节省下载时间和存储空间

（2）允许大量并发用户

（3）可以随机访问和互动操作，实时性强

（4）流媒体数据在时间上连续

基于上述特点，流媒体技术在各个领域迅速发展。随着未来 5G 技术的不断普及，流媒体技术必将有更大更广的应用空间。流媒体广泛应用于在线视频点播、电子商务、在线直播、远程医疗、视频会议、网络监控等网络信息服务。可以说，流媒体技术已经成为互联网上的主流应用。

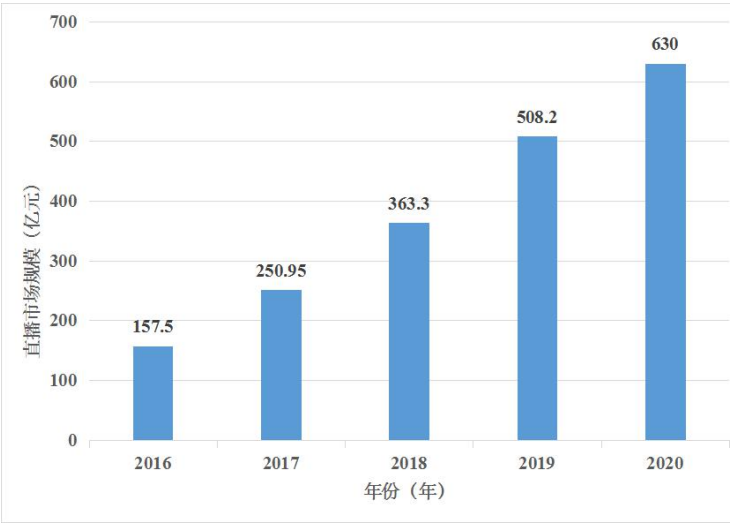


图 1.1　2016-2020 年中国直播行业市场规模及预测

Fig. 1.1　2016-2020 Market size and forecast of China's livestreaming industry

网络直播的门槛很低，从最初需要一台电脑到现在只需要一台手机就可以进行直播。数据显示，2015 年我国网络直播平台数量接近 200 个，其中网络直播市场规模约 90 亿，网络直播平台用户数量已达到 2 亿，大型直播平台在一天的高峰期，有近 400 万人同时观看，超过 3000 个房间在网上直播，近 50%的网民说他们在网上浏览过直播。

根据中国商业产业研究院《2018-2023 年中国在线直播行业市场前景及投资研究报告》[3]的统计，2017 年中国直播行业市场规模达到 250.95 亿元，同比增长 59.3%；随着移动直播的全面普及和娱乐消费的升级，预计 2018 年中国直播行业市场规模达到中国直播业的市场规模预计在 2018 年达到 363.3 亿元，增长率为 44.8%。

数据表明，2016 年中国网络直播行业在爆发式增长后，直播行业的激素经济逐渐衰退，行业开始回归理性，市场开始良性发展。2017 年，中国网络直播行业用户达到 3.92 亿，同比增长 26.5%。预计 2018 年新增用户将进一步增加到 4.52 亿，同比增长 15.3%。

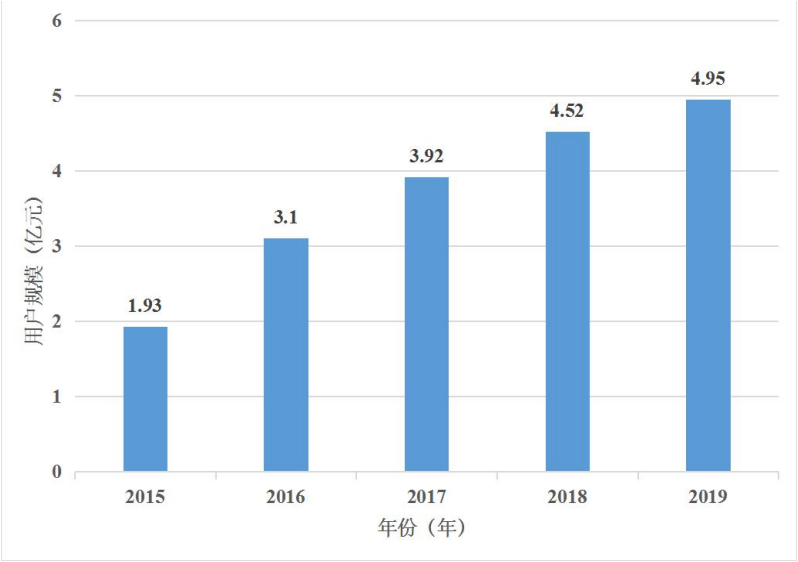


图 1.2　中国在线直播市场用户规模及预测

Fig. 1.2　China online live broadcast market user scale and forecast

基于上述背景，本文结合各种流媒体技术的优点和特点，提出了一种基于 FFmpeg 和 SDL 的智能录屏及播放系统，本系统推送视频流端使用开源方案 FFmpeg 和 DirectShow 对屏幕进行录制和压制，充分利用 GPU 运算能力进行硬件编码，之后将其发送到流媒体服务器进行负载均衡方面的处理，再由接收视频流端向服务器获取视频流进行解码，通过 SDL 进行显示。

本文设计的系统解决了传统直播方式下视频质量受网络传输抖动影响的问题，并通过性能优化提高了视频播放的实时性和流畅性，满足了网络安全监控系统和直播的需要，为以后的发展打下了基础。在安全和现场领域中具有较高的研究意义和工程实用价值。

1.2 国内外研究现状

1.2.1 流媒体播放器研究现状

流媒体技术的快速发展主要依赖于视频编码技术的不断完善。未压缩的视频数据量非常大，给网络传输和计算机存储带来很大困难，因此必须对视频进行压缩编码。目前主流的视频压缩技术主要有两个系列：ISO 的 MPEG 系列(MPEG-1、MPEG-2、MPEG-4)和 ITU 的 H.26x 系列（H.261、H.263、H.264 和未完成的 H.265）[4]。压缩水平的提升使得高清流媒体的传输越来越方便。

随着流媒体技术的发展，流媒体播放器同时也得到了迅速的发展。目前，市场上有数以百计的 PC 端流媒体播放器，其中主流的播放器有三种：

（1）VLC

VLC 原指 VideoLAN 客户端(VideoLANClient)，它是一个开源、跨平台、可扩展、多媒体播放器、流媒体服务器和框架，可以播放大多数多媒体文件，以及 DVD、音频 CD、VCD 和各种流媒体协议，现更名为 VLC media player[5]。

VLC 采用全模块化结构。通过在系统中动态加载所需的模块，可以在 module_bank 的结构中管理它。甚至 VLC 的 Maine 模块也通过插件动态加载（当初始化 module_InitBank 函数以建立 module_bank 时）。对于不支持插件动态加载的系统，VLC 也可以在 VLC 启动时使用内置函数静态加载所需的插件，并将其放入 module_bank 中进行统一管理。无论是流媒体服务器还是多媒体播放器，VLC 本质上都是一个“播放器”，处理 ES、PES、PS、TS 等流媒体之间的转换、传输和显示。

（2）Windows Media Player

Windows Media Player 是微软的免费播放器。它是微软 Windows 的一个组件，通常被称为“WMP”，可利用插件增强功能[6]。

WMP 是基于 DirectShow 架构的封闭结构媒体播放器，不过该播放器仅适用于 Windows 操作系统，而且存在响应速度慢、占用系统资源多的缺点。

（3）RealPlayer

Real Player 的主要功能包括：支持各种在线媒体视频的播放，包括 Flash、FLV 或 MOV 格式，以及在播放过程中记录视频的能力。它还向浏览器中添加在线视频，支持

Internet explorer 和 Firefox，因此您可以将 YouTube、MSN、Google Video 和其他在线视频下载到本地硬盘上进行脱机查看。还增加了 DVD/VCD 视频烧录功能[7]。

综上所述，现在主流的流媒体播放器功能已经非常完备，但是可移植性和扩展性不强，而且许多功能并不是用户真正需要的，不够简洁，功能性溢出，加之对用户的封装性差，用户使用起来不够友好。

1.2.2 FFmpeg 及其概述

通过网络传输的流媒体数据都会被编码和压缩，并且只有在解压缩后，视频数据才可以显示和播放。FFmpeg 是一种开源、免费、跨平台的软件，可用于录制、转换和流媒体视频。FFmpeg 是基于 Linux 操作系统的，但它也可以被编译并用于大多数操作系统，如 Windows，它支持超过 40 种编码，如 MPEG、DivX、MPEG4、AC3、DV、FLV，以及超过 90 种解码，如 AVI、MPEG、OGG、ASF[8]。

目前，对 FFmpeg 的研究主要集中在 FFmpeg 编码和嵌入式系统的应用。在 FFmpeg 视频转码方面，针对 FFmpeg 二次开发的具体需求，FFmpeg 逐渐应用于基于网络的分布式视频转码与分发系统中。FFmpeg 嵌入式应用主要是嵌入式平台的构建和研究。

国外对 FFmpeg 的研究比国内更深入。研究方向是将 libavcodec 类库应用于 FFmpeg 中的其他程序，将 FFmpeg 中的嵌入系统应用于 FFmpeg。例如，Hikki Orsila 介绍了 libavdec 的开发，并介绍了它在播放器开发和流媒体处理中的应用。

一般来说，国外对 FFmpeg 的研究比国内更深入，但相应的价格昂贵，产品也更复杂。

1.2.3 流媒体服务器研究现状

流媒体技术主要表现在编码器（编码技术）、播放器和流媒体服务器三个方面。目前主流的流媒体服务器主要包括：

（1）FMS（AMS）流媒体服务器介绍：

Flash Media Server (FMS)，后来更名为 Adobe Media Server (AMS)，是首款流媒体服务器产品，其公司是 Adobe，在流媒体视频和实时通信领域处于领先地位，在解决方案上，FMS 能够很快搭建一个实时流媒体和点播服务器[9]。

FMS 是一种商业产品，需付费后使用，参考价格在 4-5 万，教育用户可以有折扣。支持 RTMP/RTMPE，支持 HLS(m3u8)。

（2）SRS Overview

SRS 定位是一个可操作的互联网实时服务器集群，它寻求更完美的概念完整性和更加简易的代码实现[10]。SRS 提供了一个丰富的接入方案，将 RTMP 流连接到 SRS，包括将 RTMP 推送到 SRS，将 RTSP/UDP/FLV 推送到 SRS，并将流拉到 SRS。SRS 还支持 RTMP 流的各种转换，例如 RTMP 流的转码、流截图、转发到其他服务器、打包 HTTP-FLV 流、打包 HLS、打包 HDS、记录 FLV。SRS 具有大规模集群的关键特性，如 CDN 服务，RTMP 多级集群、VHOST 虚拟服务器、无中断服务重载、HTTP-FLV 集群和 Kafka 对接。此外，SRS 还提供了丰富的应用程序接口，包括 HTTP 回调、安全策略、HTTP　API 接口、RTMP 速度测量。

（3）Red5

Red5 的主要功能与 Macromedia 的 FMS 类似，它提供了基于 Java 的流媒体服务器的 Flash 流媒体服务[11]。它采用 Java 语言编写，采用 RTMP 作为流媒体传输协议，与 FMS 完全兼容。它具有流式 FLV 和 MP3 文件、实时录制客户端流作为 FLV 文件、共享对象、实时视频回放、远程等功能。将 FMS 替换为 Red5 后，客户端可以正常运行而无需更改。

综上所述，上述列举的目前市面上主流流媒体服务器中，部分商用或非开源，不便于深入的研究，本系统使用 Nginx 配合 RTMP 模块作为流媒体服务器，Nginx-RTMP 包括以下两个优点：

（1）Nginx 是一种轻量级的服务器框架，便于上手维护；

（2）基于 Nginx，直播点播使用一套服务器，管理起来较为容易；

所以，使用 Nginx 作为本系统的流媒体服务器部分是契合的。

1.3　本文的主要工作和内容安排

本文共分为六章。详情如下：

第一章，绪论。主要介绍近年来我国互联网技术的发展，特别是流媒体技术的飞速发展。结合我国直播市场的蓬勃发展，分析了高性能视频录屏及播放系统的研究现状和意义。论述了国内外流媒体播放器和服务器的研究现状，分析了 FFmpeg 技术的应用价值。

第二章，视频编码技术与 RTMP 传输协议简介。主要介绍了基于 FFmpeg 和 SDL 的智能录屏和播放系统设计实现过程中所涉及到的技术，如视频颜色空间、H.264 视频编码标准、FFmpeg 视频处理技术以及 RTMP 流媒体传输协议。

第三章，基于 FFmpeg 和 SDL 的智能录屏及播放系统总体设计。本章对系统的功能需求和非功能需求进行了深入的分析，并设计了系统的总体框架，包括客户端软件、服务器和硬件平台的设计。

第四章，基于 FFmpeg 和 SDL 的智能录屏及播放系统功能实现。本章首先讲解了整个系统的开发环境构建。接着详细阐述了 FFmpeg 具体采集视频数据的流程和客户端收流解码并显示播放的具体流程。

第五章，基于 Nginx 服务器的搭建与数据传输实现。主要讲述了 Nginx 服务器框架的搭建流程以及视频数据是如何实现网络传输的。并针对高并发请求情况，进行负载均衡处理，减轻服务器压力，优化传输流程。

第六章，系统测试及分析。本章对基于 FFmpeg 和 SDL 的智能录屏和播放系统进行了大量的现场测试和系统质量评估。首先，对系统的基本功能进行了测试。然后，通过与虎牙直播的对比实验，验证了系统的鲁棒性、时延和兼容性等性能，并根据测试结果对系统进行了全面质量评价。

2 视频编码技术与 RTMP 传输协议简介

2.1 视频颜色空间介绍

根据颜色的定义可知，其本质是通过人类大脑、眼睛与生活经验对光所产生的一种视觉反应，通俗解释就是，颜色是人通过大脑对光的一种主观感觉。颜色空间是在坐标系和子空间中描述颜色的。所以可以通过混合三种单色光来模拟人眼可以感知的几乎任何颜色，这就是三原色原理，而所谓的三原色也是由人体生理因素造成的[12]。

颜色空间有很多种，如 RGB、CMY、YUV 等。CMY 是用于工业印刷的彩色空间。它对应于 RGB。不同的是，RGB 来自于混合和添加光源，而 CMY 是基于被动物体对光吸收和反射的减去和混合。另外，CMY 是从油墨染料的三种原色中衍生出来的:青色、品红和黄色的首字母。

2.1.1 RGB 颜色空间

RGB 模型的基础是在 1931 年的剑桥会议上确定的。通过 CIE 国际照明委员会会议，采用数学方法推导出真正的原色，从而提供了一个新的色彩系统[13]。新配色系统采用汞弧光谱滤波技术，制备了 700 nm（红色）、546.1 nm（绿色）和 435.8 nm（蓝色）三种原色。RGB 色彩空间是指将这三种单色光组合在一起形成的三维色彩空间。在 RGB 三维模式下，任何颜色的光都可以通过不同的分量颜色混合，如公式 2.1 所示：

$$F = r[R] + g[G] + b[B] \tag{2.1}$$

其中 F 为待表示的色光，r、g、b 是三种原色的百分比，其关系为 r+g+b=1。各分量的比例系数如公式 2.2 所示。

$$\begin{cases} r = \dfrac{R}{R+G+B} \\ g = \dfrac{G}{R+G+B} \\ b = \dfrac{B}{R+G+B} \end{cases} \tag{2.2}$$

RGB 作为一种常见的色彩空间，经常应用于视频处理、设备屏幕显示等相关领域。然而随着人们对色彩显示技术研究的逐渐深入与追求的提升，RGB 颜色空间的弊端也逐渐凸显出来。在自然环境中，人类的视觉系统往往对亮度的变化很敏感，但对颜色的变化相对较慢；同时，RGB 色彩分量在显示过程中容易受到亮度的影响。当亮度降低

时，原色成分的显示效果也会降低。另一方面，RGB 变换是一种非线性变换。虽然由于非线性变换，颜色分量之间的相关性很小，但由于奇异点、计算量等各种因素的限制，实际使用时颜色分量的显示效果并不理想。

2.1.2 YUV 颜色空间

YUV 彩色空间用于当今 PAL 彩电标准中。研究模型的显示原理基于对人眼视觉成像分辨率的敏感性，对图像颜色差异与亮度差异进行分析，结果表明，人眼对亮度的高灵敏度高、对颜色的敏感度低，所以使用 Y 表示视频图像的亮度信息，通过 U 和 V 表示视频图像的颜色信息[14]。人眼的高灵敏度，即亮度信息 Y，由图像细节表示，而相对低的灵敏度 U 和 V 则由颜色渲染。一般来说，常见的 YUV 颜色空间格式具有不同的比例，例如 Y:U:V= 4:4:4，4:2:2 和 4:2:0。

在 YUV 444 格式中，每个通道具有相同的水平和垂直采样率，这保证了视频数据分量信号的完整性，因此其数据量是三种格式中最大的。在 YUV 422 格式中，紫外色差通道的采样率与 Y 采样率之比为 1:2。虽然 YUV420 格式不只表示 Y 和 U 分量，但是它主要以同一行表示，用于仅存储一个色度分量。例如，如果第一行的 YUV 信道采样率之比为 4:2:0，那么第二行的采样率之比为 4:0:2。因此，YUV 420 在三种不同采用的格式中具有最小的数据量。由于本项目对系统整体的网络延时有极高的要求，在网络带宽有限的情况下，尽可能用更少的数据量来实现流媒体数据的传输，所以本项目中使用 YUV420 作为颜色空间。

2.2 H.264 视频编码研究

表 2.1 主要视频编码标准一览表

Tab. 2.1 List of major video coding standards

名称	推出机构	出现时间	目前使用领域
HEVC（H.265）	MPEG/ITU-T	2013	未普及
H.264	MPEG/ITU-T	2003	各个领域
MPEG-4	MPEG	2001	不温不火
MPEG-2	MPEG	1994	数字电视
VP8	Google	2008	未普及
VP9	Google	2013	研发中

视频编码，是将如 RGB 或 YUV 原始视频像素数据进行压缩工作，在经过编码压缩后视频序列整体的数据量将会被大幅降低，通常情况下，如果一个视频序列在不经过编

码压缩而进行存储传输，其占有的数据空间与网络带宽将会非常的大。因此，视频编码标准的制定具有重大的意义。其次，各种视频编码标准的应用面与侧重点各不相同，下表显示目前较为主流的视频编码标准。

从表中可以看出，有两种新的视频编码方案：VP9 和 HEVC，但目前这两种编码标准由于开发时间较长，还没有达到一定的实用性和普及性。此外，MPEG 系列频率标准对本地文件存储功能具有良好的支持特性，而 H.26x 系列视频标准对网络传输特性表现出强烈的重视。因此，HD 实时网络流媒体播放系统通常采用普通的 H.264 视频编码标准。

2.2.1 H.264 标准基本框架

2001 年，国际电信联盟(ITU-T)和 IOS/IEC 这两个主要的视频标准制定组织共同成立了视频联合集团 JVT，共同开发新的视频编码标准[15]。因此，H.264 作为合作研究的成果孕育而生，继承了前代标准中的高效编码算法，并作为 MPEG-4 的第 10 部分发布，因此又被称为 MPEG-4 AVC。从技术角度来看，H. 264 标准注重对压缩编码效率和网络传输可靠性的多维度提高，具有更先进的标准设计思路和理念。因此，H.264 编码标准适应于多种应用环境与领域的需求。并且，H.264 标准中没有对编解码结构如何实现做细致硬性规定，只要满足标准中编码器输出码流语法和比特流解码格式的条件，就可以使不同编解码器模型下的 H.264 数据进行交互沟通，因此使 H.264 具有更高的环境适应性。

参考上一代的视频标准，H.264 标准在不同的应用场景下提供了不同的级别和档次。包括 Baseline Profile 基本档次、Main Profile 主要档次和 Extended Profile 扩展档次，2005 年增加了高档次 FRExt，主要用于高精度领域。不同档次就会需要不同性能的编码器，分为不同的级别，每个级别支持不同的视频编码分辨率与比特率。各等级如表所示。

表 2.2 H. 264 档次表

Tab. 2.2 H. 264 class table

档次	功能模块	主要应用
Baseline	帧内预测、帧间预测、适应性变长编码 CAVLC 等	视频会议、可视电话等低延时领域。
Main	隔行视频、B_SLICE 帧间预测、加权预测、自适应算术编码	视频存储、数字视频广播等领域。
Extended	SP 与 SI 片码流切换、误码性能改进、数据分割	流媒体应用领域。
FRExt	加入保真度范围、变量化矩阵和自适应尺寸变换。	无损领域，如数字电影

H.264 编码采用分层设计的概念，将整个编码框架分为两层：视频编码层(VCL)和网络抽象层(NAL)。VCL 层主要负责视频编解码，包括视频序列帧内预测、DCT 变换编码、熵编码等模块。NAL 层主要负责底层网络传输，包括数据打包、帧化、逻辑信道信令等功能，层次结构如图 2.1 所示。

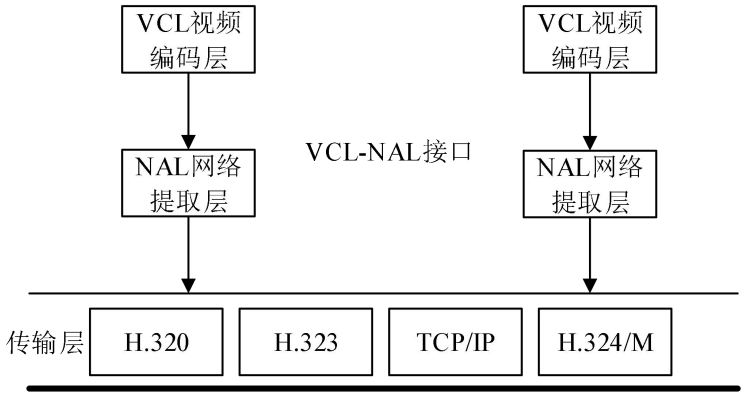


图 2.1　VCL-NAL 接口

Fig. 2.1　VCL-NAL interface

前代编码标准都以高压缩比高效率为主要目的，而随着网络技术的发展渗透，在对编码效率的提升要求外，对视频质量与网络适应性要求也在提高，此前的编码标准模式面对复杂网络环境的亲和力不足等弱点逐渐显露出来，因此，H.264 就是对编解码性能与不同网络环境扩展性能支持的增强，通过分层结构的技术模式，VCL 与 NAL 接口使得这两层在不需要对码流重规划的情况下就可以互通，并且提高了可移植性，在不同的系统里可以保证编码设计与网络接口设计不冲突，同时达到最佳的编码与网络传输性能。

2.2.2　H.264 视频编码原理

视频数据的编码压缩是基于两个前提条件，数据冗余和视觉冗余。这两种非必要数据大量占用有效信息的存储比率[16]。其中，数据冗余主要指视频图像中相同信息的重复存储，消除这些重复信息后并不会导致信息损失，因此这种压缩编码称之为无损压缩。而视觉冗余则指的是在人眼的视觉感知系统中，对色彩亮度等信号强弱变化的分辨能力不同，在一定范围内对不同信号的敏感度不同，在对视觉冗余信息进行消除后，对图像的感知变化不大。因此，可以利用此项特性对视频图像进行一定的失真编码。这种编码压缩方式属于有损压缩。

H.264 视频编码标准正是基于上述两种条件对数据进行高压缩比处理，在图像信息损失不大的情况下，大幅提高了视频数据的传输与存储效率。一般情况下，视频压缩编码采用多种编码方式混合的形式，在压缩编码的基础上，对输出数据进一步压缩处理，极大地提高了数据的压缩效率。常用的编码方式如下：

（1）变换编码

变换编码主要原理是通过不同域之间的相互转换而产生相关性较小的系数，通过对转换系数进行编码来去除视频图像之间的相关性。一般是通过空间域转换到频率域，再对转换系数进行编码处理，这样就可以实现去除相关性，且使图像能量更加集中。常见的正交变换有 DFT 变换，DCT 变换等。

视频信号经过离散余弦变换后还需要对其进行数字量化处理，在量化过程中，可以对图像的高频细节信息不做或者做少量传递，因此在量化过程中，进一步降低了数据大小。此外，数字量化过程对图像信息的丢失情况比较严重，因此是图像信息损伤的主要原因之一[17]。量化过程可以用公式 2.3 表示：

$$F_Q(u,v) = round\frac{F(u,v)}{Q(u,v)\times q} \tag{2.3}$$

其中 $F_Q(u,v)$ 表示信号量化之后的 DCT 系数，$F(u,v)$ 表示在信号量化之前的 DCT 系数，$Q(u,v)$ 表示量化加权矩阵，q 表示量化步长，$round$ 表示归整，即对得到的值进行整数取值，选择最接近的整数值。

（2）熵编码

熵编码的基本原理是通过短编码来缩短信源中符号的出现次数或概率，对于长编码来说，对于小符号的出现次数或概率，使得它能够统计地缩短平均码长。VLC 编码通常用于霍夫曼编码、算术编码、RLC 编码等。

一种更有效的方法是在量化器输出 DC 系数之后，通过 Z 扫描扫描输出 AC 系数。在二维量化系数转换成一维序列的前提下，进行 RLC 游程编码。最后，对编码数据执行另一可变长度编码。游程编码是一种简单、快速的压缩编码方法，取得了显著地效果，得到了广泛的应用。

（3）运动估计与运动补偿

通常，视频中的图像序列在小的时间间隔内变化不大，因此它们高度相关，可以通过运动估计和运动补偿来消除。时间相关反映在序列中的帧之间的相似性。在主帧变化小的视频序列中，图像表现出较强的时间相关性。为了提高编码效率和压缩比，不需要

对每帧进行单独编码和压缩。在这种情况下，运动估计和运动补偿的任务是对相邻图像帧中变化很大的部分进行编码，从而进一步减少数据量。

运动估计的逻辑是将当前的视频图像分割成几个图像块。通过比较前后两帧的图像相似度来搜索相似度最高的块。这种搜索过程称为运动估计。

通过计算两图像块之间的位置信息，得到运动矢量。然后，通过根据运动矢量信息从参考图像块中减去当前图像块来获得残余图像块。因为参考图像的选择在编码过程中非常重要，所以编码器通常将视频序列帧分成三种不同类型[18]：关键帧 I(Intra)、双向预测帧 B(Bidirection prediction)和预测帧 P(Prediction)。

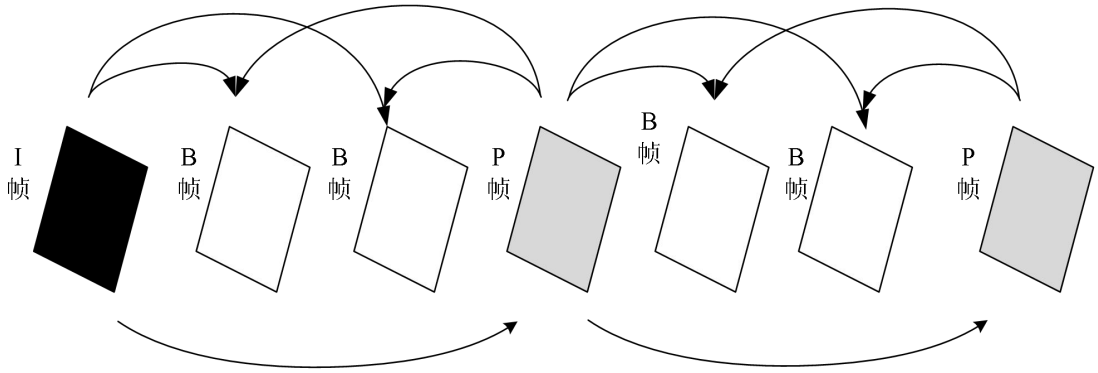


图 2.2　I、B、P 帧结构顺序

Fig. 2.2　I、B、P frame structure sequence

在图中所示的视频序列中，关键帧 I 是独立编码的，不依赖于其他图像帧。主要用于对帧内的压缩算法进行编码。因此，运动估计和运动补偿是不必要的。预测的帧 P 移动以补偿之前的关键帧 I 或预测的帧 P，然后编码残值。双向预测帧 B 通过前后方向的参考帧在帧间预测中进行双向预测。因此，与 P 帧相比，B 帧具有更高的压缩比。

（4）混合编码

如图 2.3 所示，混合编码采用 DCT 变换编码、熵编码、运动估计和运动补偿对视频图像序列进行编码，并将其组合在一起，实现了较高的编码和压缩效率。

首先，将图像分为两部分，下部分通过运动估计和运动补偿进行块分割得到块 X，上部分通过与 X 做减法得到残差图像块 X'。另一部分为 X'，通过量化和 DCT 变换的逆过程得到，然后将 X' 和 X 块相加并输出到帧存储器。

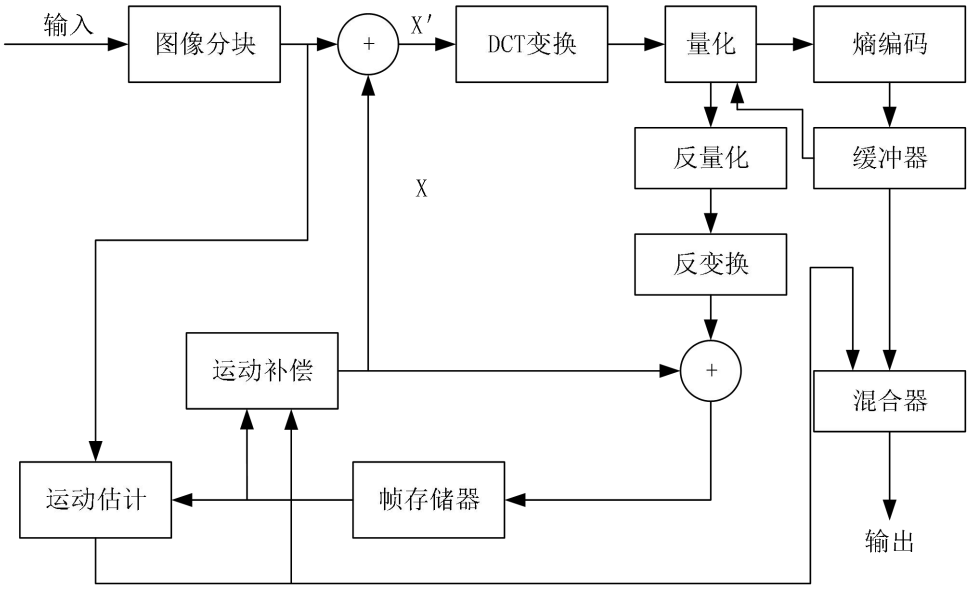


图 2.3　混合编码模型

Fig. 2.3　Hybrid coding model

2.2.3　H.264 预测技术研究

压缩编码通常是以高复杂度为代价来实现的，H.264 作为一种高效的视频编解码技术标准，继承了前代基于块匹配的混合编码方案，还引入了 va 算法和各种先进的编码技术，如基于像素 4×4 块 DCT 变换的不同模式的高精度运动补偿、帧预测和自适应的去块滤波技术。其中，帧内和帧间预测技术是大幅度提高视频序列编码效率的两项重要技术。

（1）帧内预测

以往的编码标准采用帧间预测编码，忽略了每个图像的宏块之间的相关性，使得编码后的 I 帧数据相对较大。在这方面，H.264 引入了帧内预测算法来减少图像帧的空间冗余。通过编码预测值与实际值之间的差值，大大降低了码率。对于区分亮度和色度的 H.264 编码标准，为亮度宏的块内预测模式提供 4×4 和 16×16 块大小，图像的亮度细节应该不同。当更多细节可用时，编码器使用帧预测中的 4×4 像素，16×16 方法来减少细节。对于颜色块的预测，H.264 采用 8×8 彩色块预测方法[19]。

有 9 种 4×4 亮度预测模式。其主要原理是通过参考当前像素块上方的 9 个像素和左侧的四个像素从上到下预测当前块。具体预测模式如图 2.4 所示，箭头方向表示各预测模式的方向。待预测的像素块是 a~p 块，A~M 块为相邻块中重构的参考像素块。

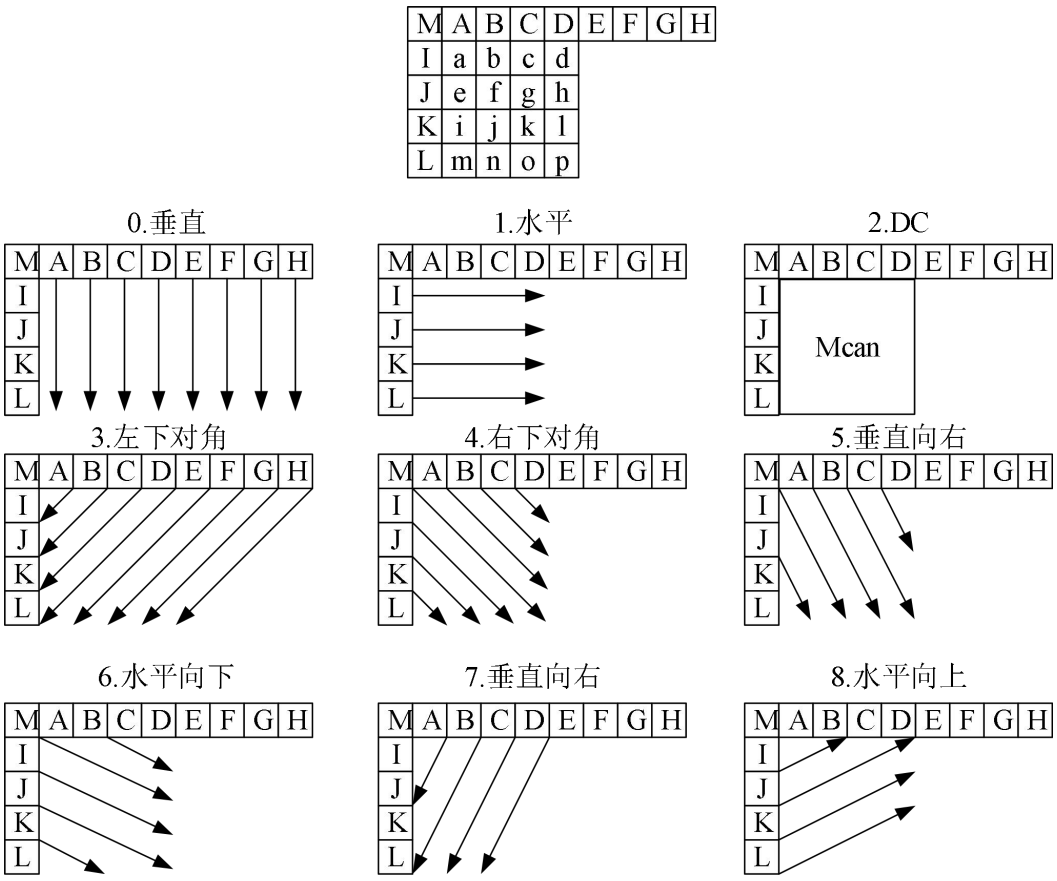


图 2.4　4×4 亮度块预测模式

Fig. 2.4　4×4 Brightness block prediction model

对于 16×16 亮度，有四种预测模式，Mode 0 垂直模式、Mode 1 水平模式、Mode 2 DC 模式和 Mode 3 平面模式。

在垂直模式下，预测宏块上方的编码像素 H 为当前宏块的列引用块。在水平模式下，宏块左侧的编码像素 H 被预测为当前宏块的行引用块。在 DC 模式下，平均宏块的左右像素 H 和 V 预计为当前宏块。在平面模式下，对宏块左侧和顶部的像素值进行插值，得到预测值，如图 2.5 所示。

对于色度信息只有亮度信息一半的 4:2:0 采用视频序列，通过预测左侧和上侧色度块，得到 8×8 色度块预测模型中的宏块。在 16×16 亮度预测中的其他四种预测方法类似于 8×8 色度块预测。

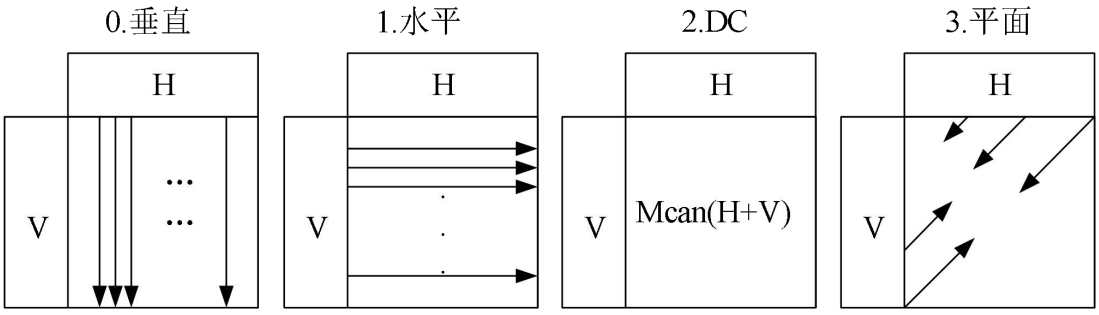


图 2.5　16×16 亮度块的 4 种预测模式

Fig. 2.5　16×16 Four prediction modes of brightness blocks

（2）帧间预测

帧间预测的基本思想就是通过参考关键帧或预测帧对当前帧进行差值预测编码，从而达到对时间上的相关性与冗余信息进行消除的目的。H.264 标准在之前的编码框架基础之上，采用了更高精度的 1/4 像素运动搜索、可变大小的参考帧预测、多参考帧预测等技术，进一步提高了编码性能。

预测块的可变大小是指将每个宏块分割为不同大小的子块。分割格式的大小分别为 16×16，16×8，8×16，8×8，8×4，4×8，4×4。在运动估计中，通过运动搜索算法对每个块模式的成本进行比较，以选择最佳的块模式。这种宏块分割方法通过适当的划分，改善了宏块之间的相互作用，提高了运动估计的精度。其缺点是编码算法的压缩性能会受到些许影响。分块如图 2.6 所示。

实验结果表明，1/4 亚像素精度基本达到了提高运动估计性能的极限。再提高精度不仅不能显著改善编码算法，而且会产生较高的计算复杂度。因此，H.264 编码使用了亮度分量的 1/4 和 1/2 的运动估计，通过插值 1/2 像素得到 1/4 亚像素[20]。1/2 像素点 Q 插值如式 2.4 所示。

$$Q = round((A - 5 \times B + 20 \times C - 5 \times E + F) / 32) \tag{2.4}$$

其中，A、B、C、D、E、F 分别代表相邻 6 个垂直整像素。

1/4 像素点 P 插值如式 2.5 所示。

$$P = round((C + Q) / 2) \tag{2.5}$$

其中，C 为半像素点中的整像素，Q 为水平半像素点。

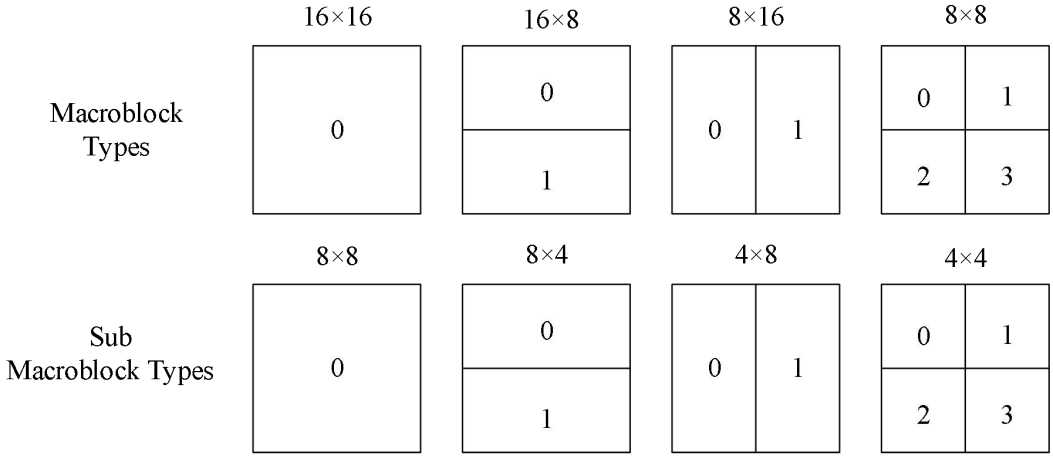


图 2.6　H.264 预测块可变大小模式

Fig. 2.6　H.264 predicts a block variable size model

2.3　FFmepg 视频处理技术介绍

FFmpeg 是一个开源的 Linux 项目，具有多种视频处理方法。它在视频处理领域具有领先的库编解码能力。它也是各种工程应用中广泛使用的编码和解码解决方案。如使用 FFmpeg 作为内核播放器的 Mplayer、storm 视频、QQ 视频、KMPlayer 等[21]。实际上，FFmpeg 的视频处理能力非常强大，几乎覆盖了所有现有的视频编码标准。因此，只要与视频处理技术的发展相关，就很难与之分离。此外，FFmpeg 不仅在视频处理方面具有很高的效率，而且具有很好的扩展性，可以在多个平台上进行编译和移植。主要模块包括：

ibavcodec：包括 avcodec 视频编解码器、DXVA2 （DirectX 视频加速）视频硬件加速规范、VDA 硬件加速、XVMC 等功能模块。它是 FFmpeg 类库的核心。它实现了大部分编解码器函数，并得到了广泛的应用。

libavformat：提供了视频多路复用封装、字幕流、视频分离等处理技术的通用框架，包括对编码解码所需的上下文结构信息的访问、流媒体信息的访问、多路复用 Muxer 和反多路复用 Demuxer 的访问，同时支持对多个媒体资源的输入和输出访问。

libswscale：Swscale 类库用于像素数据的格式转换、颜色映射转换、图像尺寸拉伸、缩放显示等。

libavdevice：AVDevice 可以读取多媒体设备数据用于硬件采集、加速，或者将数据输出到指定的多媒体设备、存储器和本地文件。

libavutil：包括一些内存的操作、CRC(Cyclic Redundancy Check)循环冗余校验、内存分配管理、大小端序列转换等功能。

libavfilter：AVFilter 可以给视频添加多种滤镜，例如在视频播放过程中添加水印效果。

由于 FFmpeg 对多媒体数据强大的处理能力以及对下一小节所要讲述的 RTMP 协议的完美支持，所以本项目参考 FFmpeg 源码，对复杂冗余的功能进行了裁剪，优化了录屏、编解码、推流以及收流播放流程，使系统更加轻量化，不仅增强了鲁棒性，也同时提高了代码可读性，便于维护以及之后功能的添加。

2.4 RTMP 的流媒体传输协议研究

RTMP 是 Real Time Messaging Protocol 的缩写，它是由 Adobe 开发的用于在 Flash 平台和流媒体系统之间传输视频数据的私有协议[22]。由于 Flash Player 在全球近 99%的安装率，使用 RTMP 技术的流媒体系统将显示出非常明显的优势，也就是说，使用 Flash Player 作为实时观看客户端将不需要安装任何插件，打开相应的实时网页即可观看直播，因此，它的便捷性被许多直播系统所采用。

业界主流的直播协议除 RTMP 之外还有一种协议：HLS(HTTP Live Streaming)，HLS 的基本原理就是当采集推流端将视频流推送到流媒体服务器时，服务器将收到的流信息每缓存一段时间就封包成一个新的 ts 文件，同时服务器会建立一个 m3u8 的索引文件来维护最新几个 ts 片段的索引。当播放端获取直播时，它是从 m3u8 索引文件获取最新的 ts 视频文件片段来播放，从而保证用户在任何时候连接进来时都会看到较新的内容，实现近似直播的体验。在本项目的准备阶段时，同时实现了两个协议的直播以进行对比，通常 HLS 直播延时会达到 20-30 秒，而高延时对于需要实时互动体验的直播来说是不可接受的。其次 HLS 基于短连接 HTTP，HTTP 是基于 TCP 的，这就意味着 HLS 需要不断地与服务器建立连接，TCP 每次建立连接时的三次握手、慢启动过程、断开连接时的四次回收都会产生消耗。相对于 HLS，使用 RTMP 实现的系统延时较小通常在 1 秒左右，并且基于 TCP 长连接不需要多次建连，所以本项目选择使用 RTMP 协议进行开发。

2.4.1 RTMP 协议研究概述

RTMP 协议是应用层协议，因此 RTMP 块流不提供优先级别或类似可靠性控制来保障信息传输，因此通常配合 TCP 这样的可靠传输层协议来建立连接，RTMP 块流确保跨流的所有消息都可以按时间戳顺序传输。建立传输层连接后，RTMP 客户端和服务器端需要“握手”建立基于传输层的网络连接，主要用于传输例如 SetChunkSize、SetAckWindowSize 等控制信息。随后通过 CreateStream 命令会创建一个网络流

NetStream 连接，用于传输具体的视频数据和控制这些信息传输的命令信息。连接示意如图 2.7 所示。

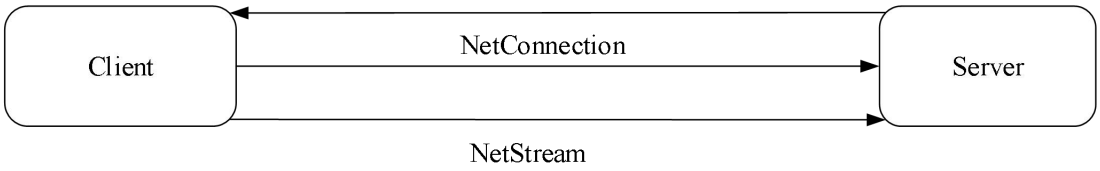


图 2.7　RTMP 流连接图

Fig. 2.7　RTMP flow connection diagram

上图中 NetConnection 是服务器程序和客户端应用程序之间的网络连接通道，NetStream 是视频数据网络传输的消息通道。虽然客户端和服务器之间只有一个网络连接通道，但是可以在这个连接通道上建立多个流媒体数据网络。

2.4.2　RTMP 的三次握手

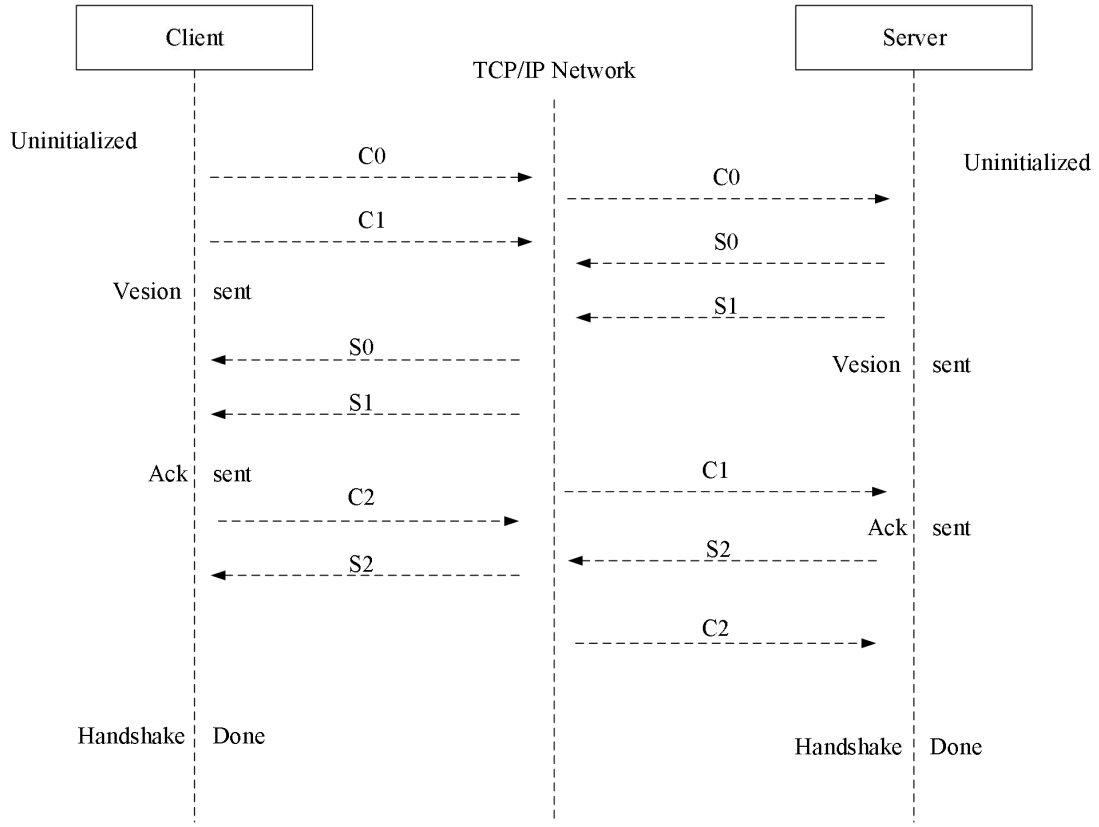


图 2.8　三次握手示意图

Fig. 2.8　Three handshake diagrams

以握手为起点建立 RTMP 连接，双方发送三个固定大小的数据块，连接成功后，可有效进行数据传输。虽然 RTMP 协议本身并没有严格指定这些消息块的具体发送顺序，但是在实现过程中仍然需要注意：

（1）握手开始于客户端发送 C0、C1 块。

（2）服务器收到 C0 或 C1 后发送 S0 和 S1。

（3）当客户端收齐 S0 和 S1 后，开始发送 C2。

（4）当服务器收齐 C0 和 C1 后，开始发送 S2。

（5）当客户端和服务器分别收到 S2 和 C2 后，握手完成。

以上的发送顺序主要目的是在保证“握手”身份验证的基础功能上尽量减少通信的次数，这一点可以通过 wireshark 抓 FFmpeg 推流包进行验证。

2.4.3 RTMP Message 消息

消息是 RTMP 协议中数据的基本单元。当 Server 端与 Client 端通信时，该消息可以包含音视频、数据或其他信息。因此，不同的消息类型具有不同的 ID，代表不同的功能。

Message Type ID 1~7：这些消息用于协议控制，一般由 RTMP 协议自身管理，用户无需操作。

Message Type ID 8~9：这些信息用于音频和视频传输。

Message 消息的报文结构如图 2.9 所示：

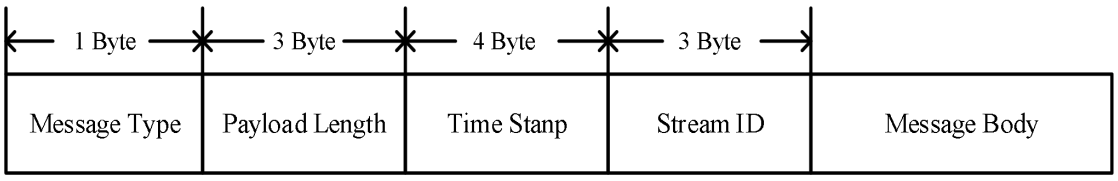


图 2.9 消息报文结构图

Fig. 2.9 Message structure diagram

其中，消息头包括以下内容：

Message Type：消息类型，用于表示消息类型。

Payload Length：负载长度是大端模式，它表示负载信息的字节数。

Time Stamp：消息的时间戳信息，以 4 个字节大端方式打包。

Stream ID：每个消息的唯一标识，分割 Chunk 和收集 Chunk 合成 Message 的时候都需要此 ID 信息来分辨是否为同一消息或消息流。以 4 字节小端存储。

Message Body：消息的负载数据。

2.4.4 RTMP Chunk 分块

Chunk 块流是 RTMP 协议中数据传输的基本单元。分区的主要意义是使高层协议的大消息被划分为小消息，以确保消息可以顺利地传输。对于数据量不是很大的消息，可以使用 Chunk Msg Header 字段来进行压缩。

在消息分块过程中，Message Body 被分成几个最大容量为 128Byte 的块，并将相应的头部添加到块的头部以形成消息块。消息分块过程如图 2.10 所示。

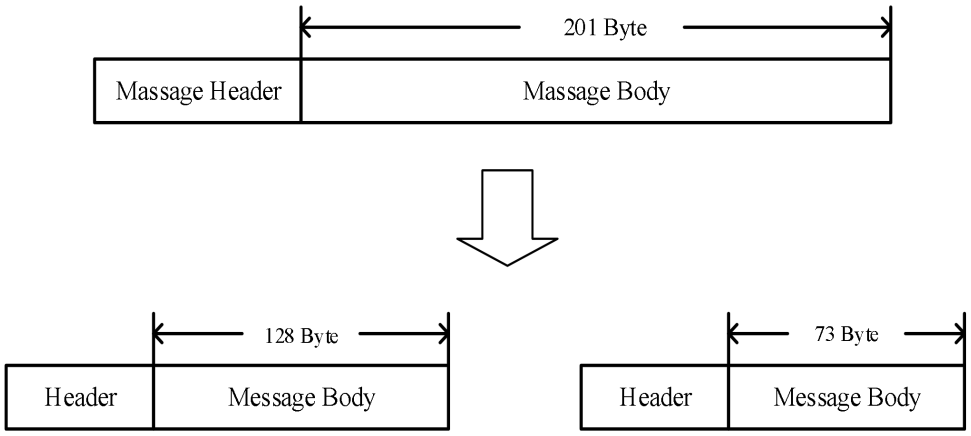


图 2.10 消息分块图示

Fig. 2.10 Message block schema

RTMP 握手建立连接后，消息（Message）被划分为以 Chunk 为单位的数据单元，所有创建的块都与唯一的 Chunk 流 ID 相关联，形成一个 Chunk 流，所以 Chunk 流其实相当于是数据传输管道，携带了一个消息流的信息。在发送时，规定只允许同时发送一个 Chunk。在接收端，每个 Chunk 都被收集为基于 ID 的消息。Chunk 共包含四个部分：

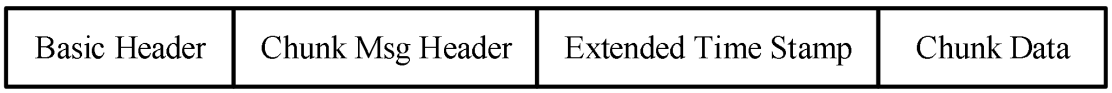


图 2.11 RTMP chunk 基本构成

Fig. 2.11 RTMP basic constitution of chunk

（1）Basic Header：基本的头信息，用于定义 chunk 类型和 ID，1~3 字节。

（2）Chunk Msg Header：块消息头编码要发送的消息信息，有 0、3、7、11 字节四种长度，长度和格式取决于 Basic Header 字段中 chunk type 和 fmt。共有四种不同的

格式：timestamp 时间戳、message length 消息数据长度、message type id 消息类型 id 和 msg stream id 消息的流 id。

（3）Extended Time Stamp：当普通时间戳的值为 0xffffff 时发送扩展的时间戳字段，而当普通时间戳的值小于 0xffff 时则不需要显示该字段。应该注意的是，扩展的时间戳存储的是完整的值，而不是差值。

（4）Chunk Data：块数据字段，发送与协议无关的数据。

2.5 SDL 跨平台多媒体开发库介绍

SDL（Simple DirectMedia Layer）是一套开放源代码的跨平台多媒体开发库，使用 C 语言写成。SDL 提供了数种控制图像、声音、输出入的函数，让开发者只要用相同或是相似的代码就可以开发出跨多个平台（Linux、Windows、Mac OS X 等）的应用软件。SDL 的设计理念为以精简的方式来完成基础的功能，它大幅度简化了控制图像、声音、输入输出等工作所需撰写的代码，目前多用于开发游戏、模拟器、媒体播放器等多媒体应用领域。

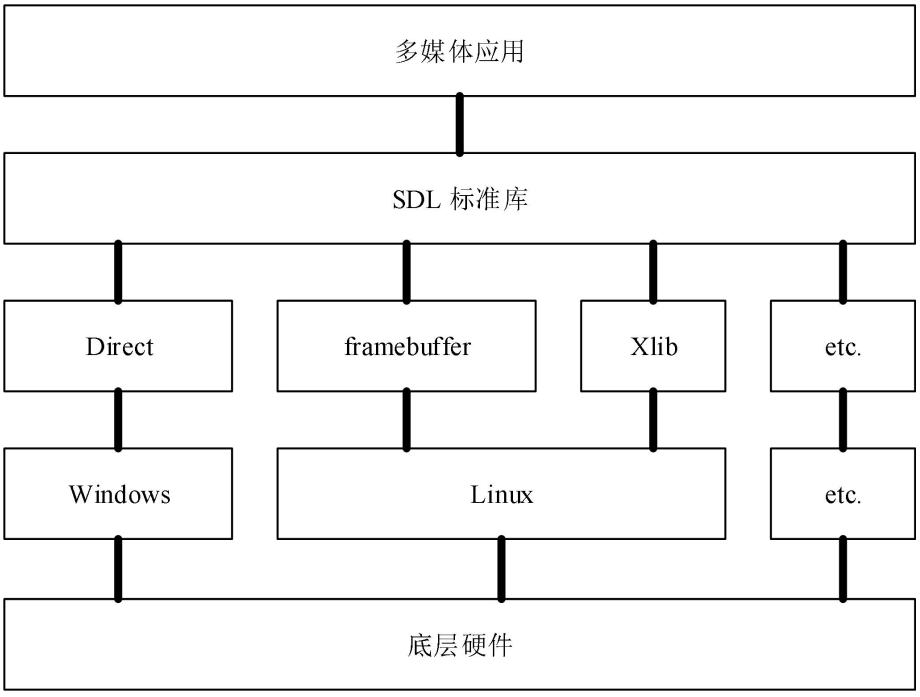


图 2.12　SDL 底层原理图

Fig. 2.12　SDL bottom schematic diagram

SDL 库分为 Video、Audio、CD-ROM、Joystick 和 Timer 等若干子系统，除此之外，还有一些单独的官方扩充函数库。标准库如下：

（1）SDL_image：支持时下流行的图像格式：BMP、PPM、XPM、PCX、GIF、JPEG、PNG、TGA。

（2）SDL_mixer：更多的声音输出函数以及更多的声音格式支持。

（3）SDL_net：网络支持。

（4）SDL_ttf：TrueType 字体渲染支持。

（5）SDL_rtf：简单的 RTF 渲染支持。

在本项目中，SDL 主要用于在收流端显示视频流，SDL 本身能做的事情比较少但结合 FFmpeg 与 SDL 后，可以发挥强大的功效。FFmpeg 每成功解码一帧便交给 SDL 并复制给渲染器，渲染器再显示出来，以此循环。这样实现的收流端有非常快的启动速度，且实现代码短小而精悍，直线式的程序流程有着非常完美的可读性，播放效果极佳。

3 基于 FFmpeg 和 SDL 的智能录屏及播放系统总体设计

3.1 系统需求分析

基于 FFmpeg 和 SDL 的智能录屏及播放系统的目的是实现多媒体数据的高清晰度实时共享。所以系统应该具备采集桌面视频数据的功能。采集的原始视频数据应采用视频编码解码技术进行解码、编码、复用和封装。最后，搭建一个流媒体服务器，将压缩编码的流媒体数据传输上去并转发，实现智能录屏及播放系统。

此外，针对现有直播系统中所存在的不足，应设计出解决的思路与方法。例如对更高分辨率的流媒体的支持；在对视频处理过程中，在编解码流程上对系统的处理效率与稳定性进行提高；在网络延时较高的问题上，采用逐帧发送与逐帧转发等方式提高直播系统的实时性等等。需求分析图如下图所示：

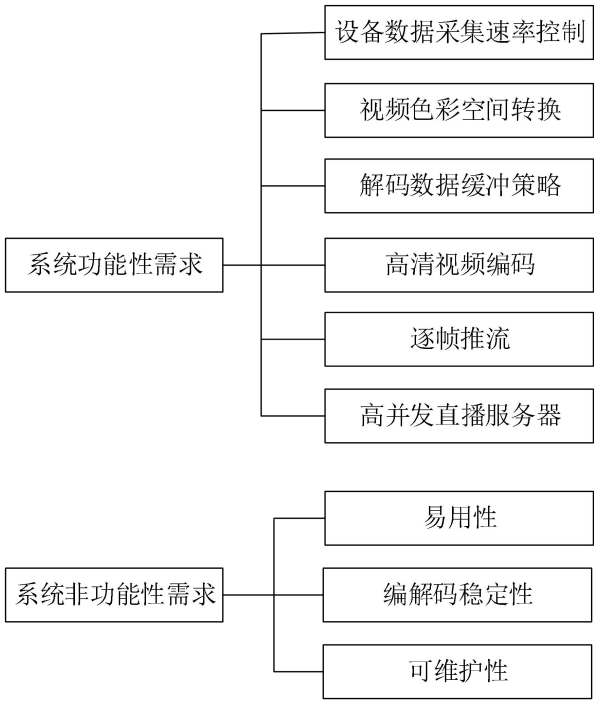


图 3.1 系统总体需求分析

Fig. 3.1 The system overall demand analysis

3.1.1 系统功能性需求

本文所设计实现的高清实时直播系统针对现有直播系统中存在的一些问题，对系统提出以下几点功能性需求。

（1）设备数据采集速率控制

在 Windows 平台下对设备的数据采集主要为计算机桌面画面数据。在数据采集过程中，需要对采集的速率进行控制，速率过高或者过低都将会影响整个系统的效率和视频质量。

（2）视频色彩空间转换

主流的色彩空间共有两种：RGB 色彩空间和 YUV 色彩空间，其中在图像或视频处理中比较常见的是 RGB 色彩空间。然而，由于人眼感官对亮度敏感，其主要颜色分量会因亮度不同而导致误差，因此显示效果并不理想。基于此，YUV 色彩空间用 Y 表示亮度信号，而 U 和 V 的色差信号改善了显示问题，因此被彩色电视标准所采用。在采集的 RGB 显示图像数据的 YUV 颜色转换之后，直播系统进行 HD 编码压缩。

（3）解码数据缓冲策略

解码数据缓冲区的设计实现，提高了系统软件在采集数据处理过程中的流畅度与稳定性，消除了多线程同步、视频解码等问题的影响。对于经过颜色空间转换后的 YUV 视频数据，写入到视频缓冲区，在视频编码线程中通过读取访问获取到数据后进行编码压缩。

（4）高清视频编码

原始视频数据一般比较大，所以编码和压缩技术也就应运而生。目前，视频编码标准分为两大类，一种是 H. 26x 系列高清晰度视频编码标准，另一种是 MPEG-x 系列视频编码标准。相比之下，H. 26x 系列具有更优秀的网络传输特性和更高的压缩效率，所以 H. 264 应该作为 HD 视频编码标准。

（5）逐帧推流

目前大多数直播系统所采用的是流媒体数据按块传输的设计，因此极大地增加了直播延时。对此，本文采用逐帧传输的设计，通过传输速率控制算法，将数据帧依次传输到服务器端，保证了系统时延可控。

（6）高并发直播服务器

系统在直播过程中对于高并发请求的处理需要高效的直播服务器支持，同时在网络延时问题的处理上，应选择实时性较好的流媒体传输协议。因此系统采用基于 Nginx 的

RTMP 流媒体直播服务器，不仅提高了直播时的最大请求连接数量，同时也更进一步地提高了系统的实时性。

3.1.2 系统非功能性需求

直播系统的非功能性需求主要体现除功能性需求以外的其他特性，包括系统的易用性、稳定性与界面美观等，具体说明如下：

（1）易用性

直播系统的易用性表现在软件安装布置简单，容易上手。在不具备视频专业基础知识的用户在使用时，提供默认的参数设置。同时在客户端设计中，强调便捷性的设计思想，可以不用重新安装客户端插件等。

（2）编解码稳定性

系统在数据采集、编解码过程中对硬件不可避免的造成高负荷，为提高采集帧率与编解码效率，应利用多线程编程技术。在对数据的处理流程中，使用多线程可以提高工作效率，同时，线程间同步技术让各线程可以稳定安全的完成任务。

（3）可维护性

可维护性是一个程序质量的重要评定，由于系统可能会进行频繁的重构，不断加入新的功能，为了保证每次对代码的维护所付出的时间成本相对较低，则必须要考虑代码的可维护性。在软件实现过程中，使用较为通用的匈牙利命名法，根据函数、变量、常量等相应的用途添加合适的前缀名或后缀名。这使得管理人员可以更方便地维护系统，同时有利于将来的扩展开发。

3.2 系统总体框架设计

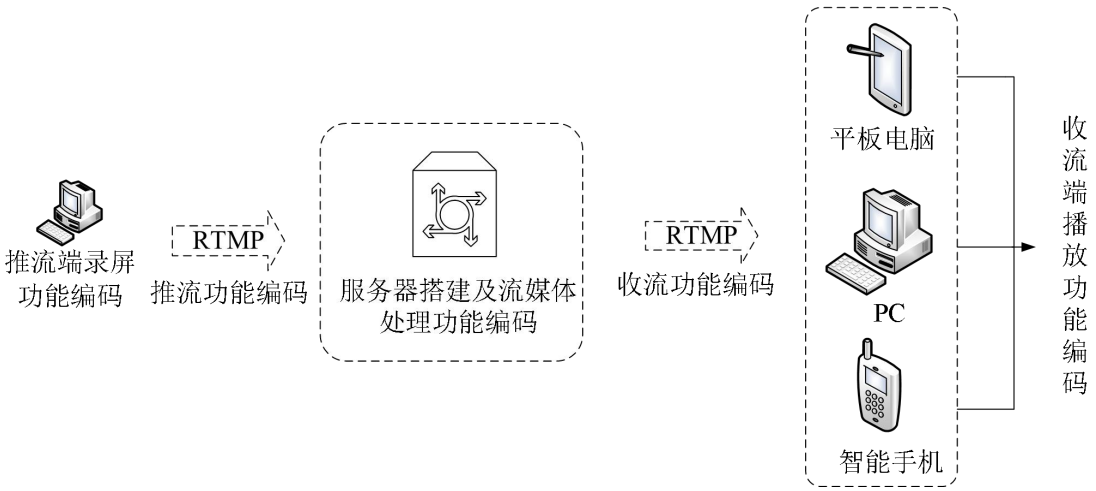


图 3.2 系统总体拓扑结构

Fig. 3.2 The overall topology of the system

本文设计的智能录屏及播放系统的组成大致可以分为三个部分：录制屏幕端（推送视频流端）、流媒体服务器和接收视频流端。录制屏幕端的软件设计采用基于 Windows 平台的 MFC 应用框架，完成视频数据的采集、编码和解码，并将视频流推送到流媒体服务器。服务器采用搭配 RTMP 模块的 Nginx 流媒体服务器实现数据的接收和转发功能。收流端从服务器获取视频流并使用 SDL 实现视频的播放。系统的总体拓扑结构如图 3.2 所示。

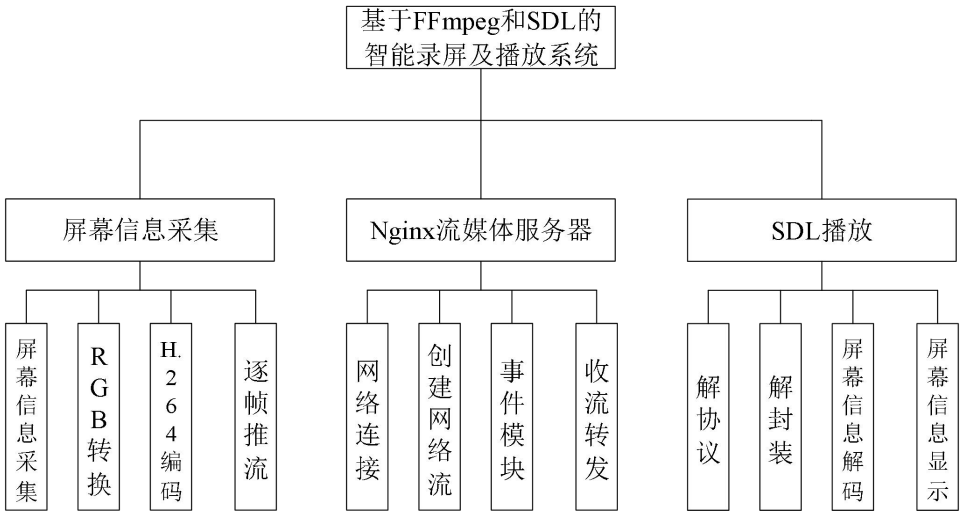


图 3.3　系统总体架构图

Fig. 3.3　System architecture diagram

系统开始工作后，录屏端将 PC 桌面录制下来进行编解码处理，通过流媒体逐帧推送到服务器。服务器将收到的数据发布到相应的直播地址，完成直播工作。系统总体架构图如 3.3 所示。

从系统架构图中可以看出，视频处理工作在推送视频流中实现，通过多线程编程技术，使软硬件资源得到充分利用。同时，在流媒体服务器端，搭建了基于 Nginx 的流媒体直播服务器，使用 RTMP 流媒体传输协议，将处理封装后的视频数据进行了网络转发。

3.2.1　客户端软件设计

客户端软件在视频数据处理中采用多线程编程，将视频捕获、视频解码和视频编码封装的功能划分为多个线程，利用线程之间的关键区域资源对线程进行同步。在采集功能中，通过控制采集速率，实现了硬件资源与视频质量的平衡。在解码功能中，通过设计解码缓冲器，提高了编码和解码的效率和稳定性。最后，通过 RTMP 逐帧地推送视频

数据，与服务器建立连接通道后，通过控制传输速率完成传输。软件设计流程图如图 3.4 所示。

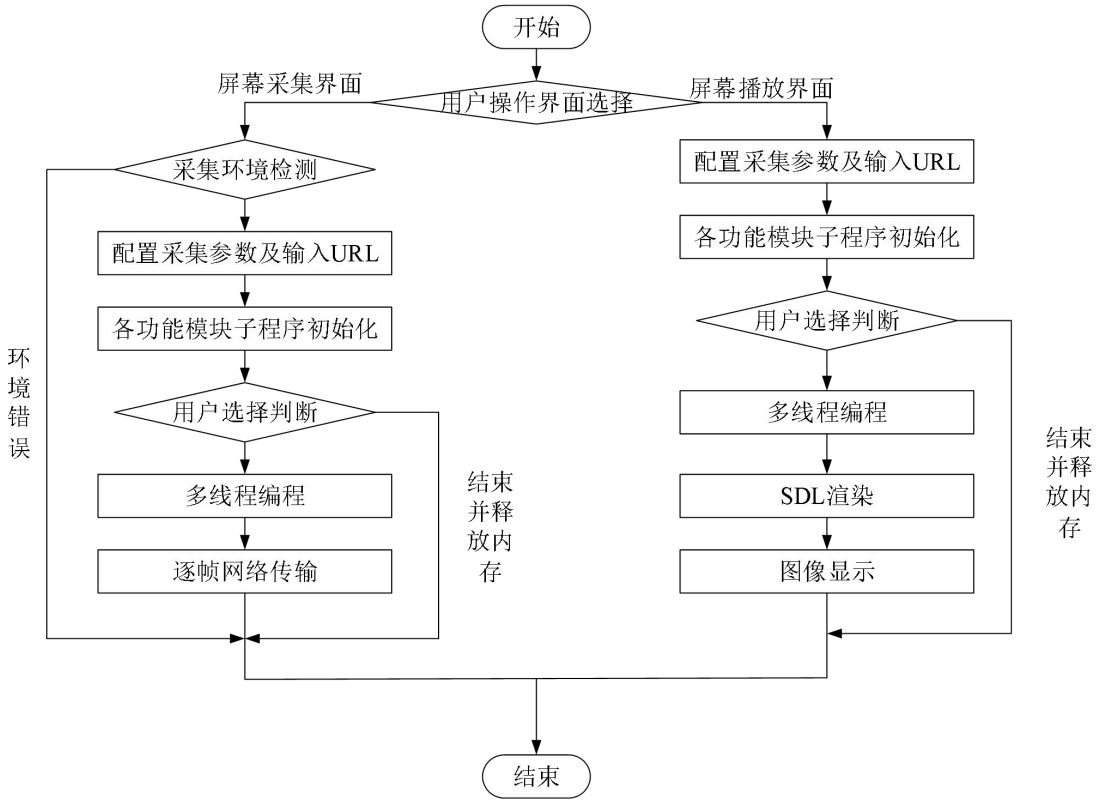


图 3.4　客户端工作流程

Fig. 3.4　Client workflow

在进行功能操作之前，客户端软件在界面顶部的菜单栏中有一个用户标识区域。可选择投影或播放以进入相应的功能界面。进入投影界面，首先列举系统中安装的采集设备，检查系统环境是否适合采集屏幕信息，能否满足采集环境的要求，最后设置采集参数并输入 URL。对于普通用户，可使用系统提供的默认参数。然后初始化各个模块以准备屏幕捕获。最后，根据用户的选择，如启动屏幕投影、停止屏幕投影、暂停屏幕投影来实现相应的功能。选择打开屏幕投影后，进行相应的屏幕信息采集、压缩编码、实现数据封装，并将封装数据流发送到流媒体服务器。

3.2.2　流媒体服务器设计

当客户端软件发送直播连接请求时，它通过直播 URL 与流媒体服务器建立连接。建立连接后，服务器在创建 live channel 后将命令消息发送到 live streaming software 完成

回复。在直播时，服务器则会将接收到的流媒体数据帧不停地转发给客户端。直播结束后，服务器收到结束消息后，清除数据通道，删除直播通道。

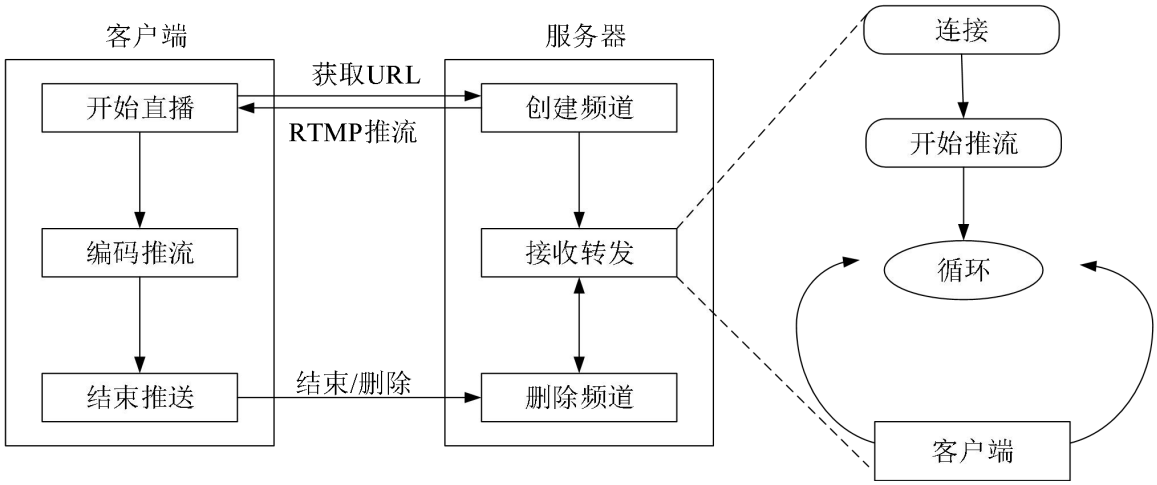


图 3.5　服务器端工作流程

Fig. 3.5　Server-side workflow

3.2.3　硬件平台

系统采用 Raspberry Pi 3B 作为服务器的硬件平台，在其 SD 卡中烧写 Centos 系统并搭建 Nginx 服务器，负责视频数据的接收、缓存和转发。Raspberry Pi（中文名为“树莓派”）是由“Raspberry Pi 基金会”设计并开发的一种基于 Linux 的卡式计算机[23]。它被广泛应用于教育及核心计算机功能的开发领域。用户还可以使用 Raspberry Pi 来编辑文档、浏览视频、绘制图片，并且由于它低廉的价格，可以被更广泛的使用。因此，许多研发部门都使用树莓派进行多媒体相关研究和开发。

图 3.6　树莓派 3B 开发板

Fig. 3.6　Raspberry Pi 3B

如图 3.6 所示，Raspberry Pi 3B 版配备博通(Broadcom)ARM 架构 4 核 1.2GHz BCM2837 处理器，1G DDR2 内存，系统存储选择使用 SD 卡。配有以太网接口、4 个 USB 接口、HDMI 高清视频输出接口，另外，还支持 4.1 版本蓝牙及无线传输。

使用 Raspberry Pi 设计的基本流程如下：首先，通过访问 Raspberry Pi 的官网，在指定的界面上下载开源的 Raspberry Pi 系统资源到本地电脑;再将准备好的系统烧入 SD 卡，即可安装系统。

4　基于 FFmpeg 和 SDL 的智能录屏及播放系统功能实现

4.1　开发环境搭建

基于 FFmpeg 和 SDL 的智能录屏及播放系统的开发环境为 Windows 64 下，FFmpeg 4.0.2 版本和 SDL 2.0.8 版本，编译环境为 Visual Studio2010（以下简称 VS2010）。

4.1.1　FFmpeg 在 VS2010 下的快速配置

（1）从 FFmpeg 官网下载 FFmpeg 库文件并下载与计算机匹配的版本：

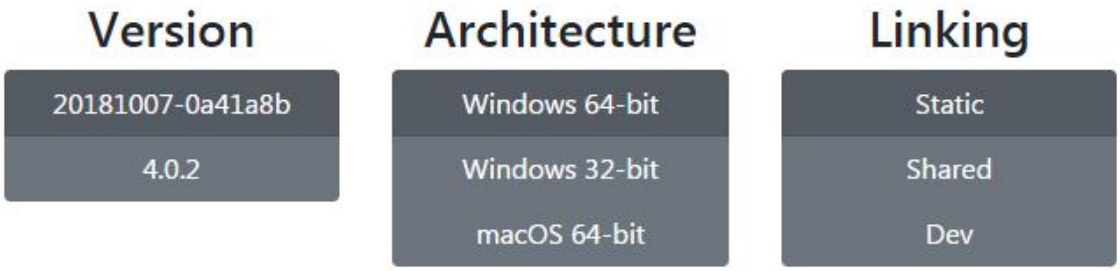


图 4.1　FFmpeg 下载版本选择

Fig. 4.1　FFmpeg download version selection

选择 Static 和 Dev 或 Shared 和 Dev 进行下载，之后为其配置 bin 文件夹。Static 版本的 bin 文件夹内存较大，而 Shared 版本的 bin 文件夹内存相对小一些，其中包含 ffmpeg.exe，ffplay.exe 和 ffprobe.exe 的动态链接库文件[24]。

（2）FFmpeg 在 VS2010 中的配置

选择 Shared 和 Dev 版本进行下载并解压，复制 Shared 版本中的 bin 文件夹和 Dev 版本的 include 和 lib 文件夹，并粘贴至新建的 ffmpeg 文件夹中。

图 4.2　FFmpeg 关键文件夹

Fig. 4.2　FFmpeg key folders

（3）环境变量的配置

右键单击计算机->属性->高级系统设置->环境变量->系统变量（所有用户都可以使用）或管理员用户变量（只有这个用户可以使用）。在系统变量中拖动滑块框，找到变量路径，单击 Edit，将路径添加到 bin 文件夹中，注意需要和前面的路径用“;”分隔。

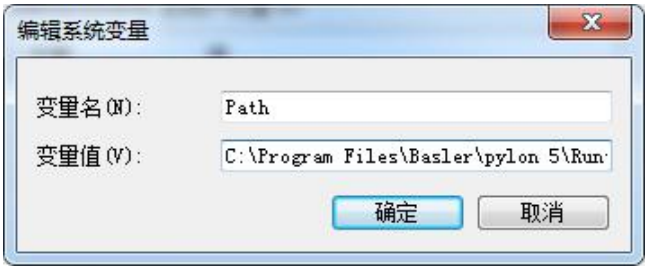


图 4.3　FFmpeg 环境变量配置

Fig. 4.3　FFmpeg environment variable configuration

（4）VS2010 的配置

打开 VS2010 编译环境，并新建一个 Win32 控制台应用程序的空白工程。

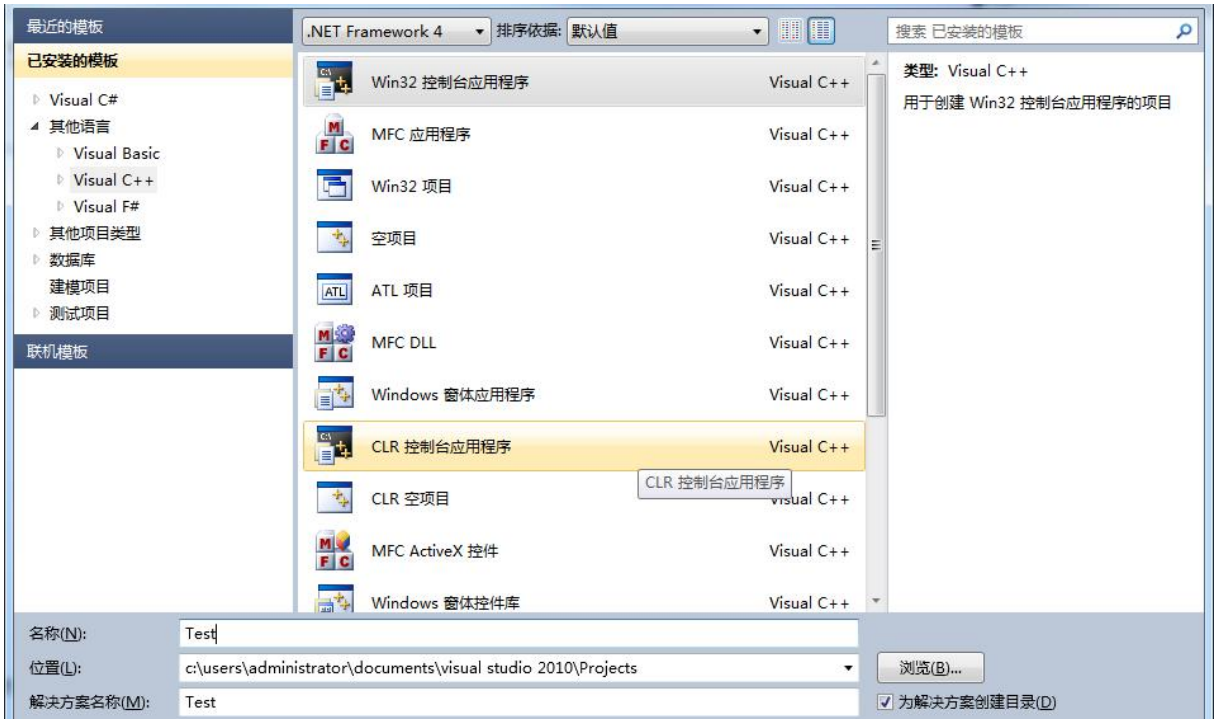


图 4.4　新建 Win32 控制台应用程序的空白工程

Fig. 4.4　Create blank project for the Win32 console application

在项目中查找属性管理器并添加名为 FFmpeg 的新项目属性表：

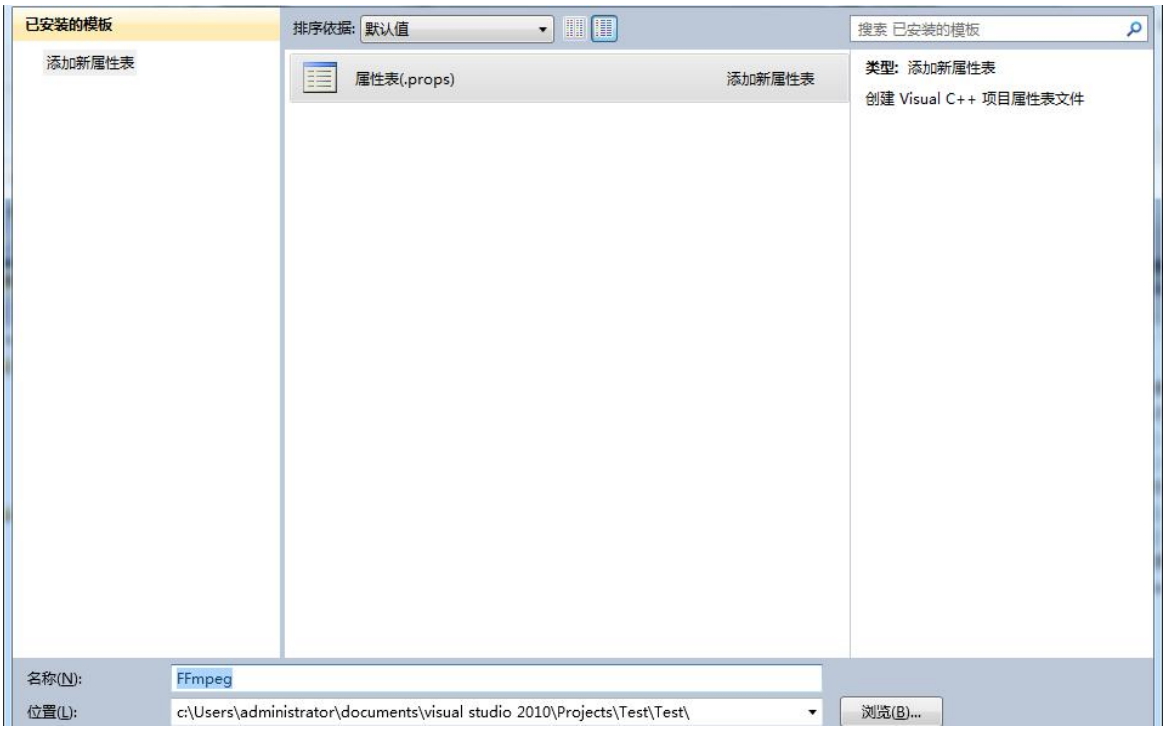


图 4.5　添加新属性表

Fig. 4.5　Add new property sheet

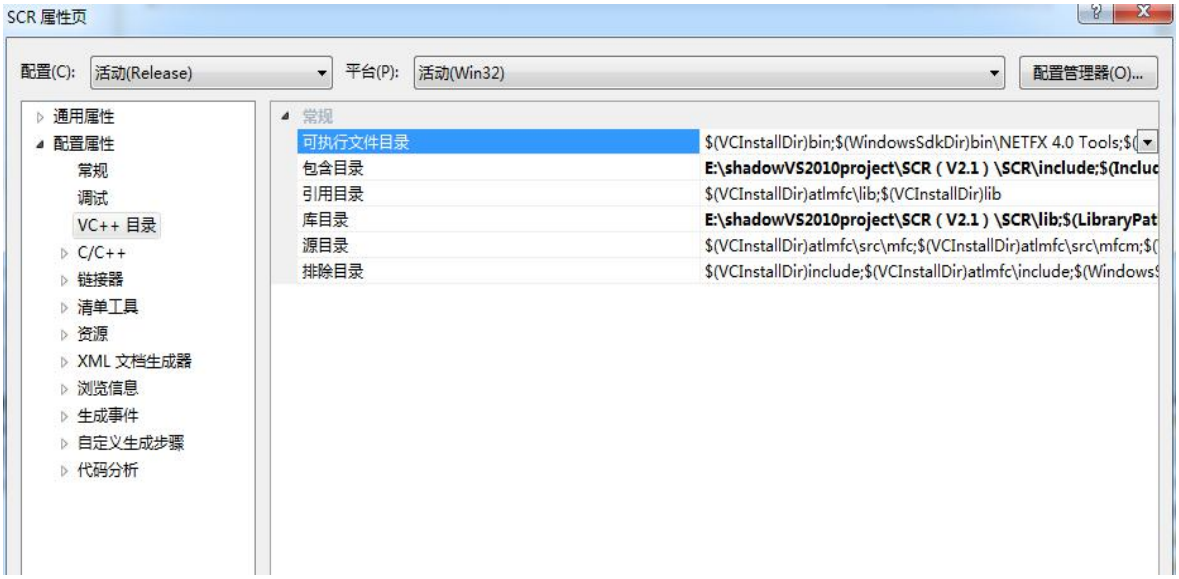


图 4.6　添加库目录

Fig. 4.6　Add library directory

选择新创建的 FFmpeg 属性表，右键单击属性，选择 VC++目录，将 FFmpeg 文件中的 include 文件夹和 lib 文件夹分别添加到 include 目录和 lib 目录中，最终结果如图 4.7 所示：

之后选择链接器->输入->添加依赖项目，将依赖的静态库文件分别手动添加进去，单击确定，得到如下结果：

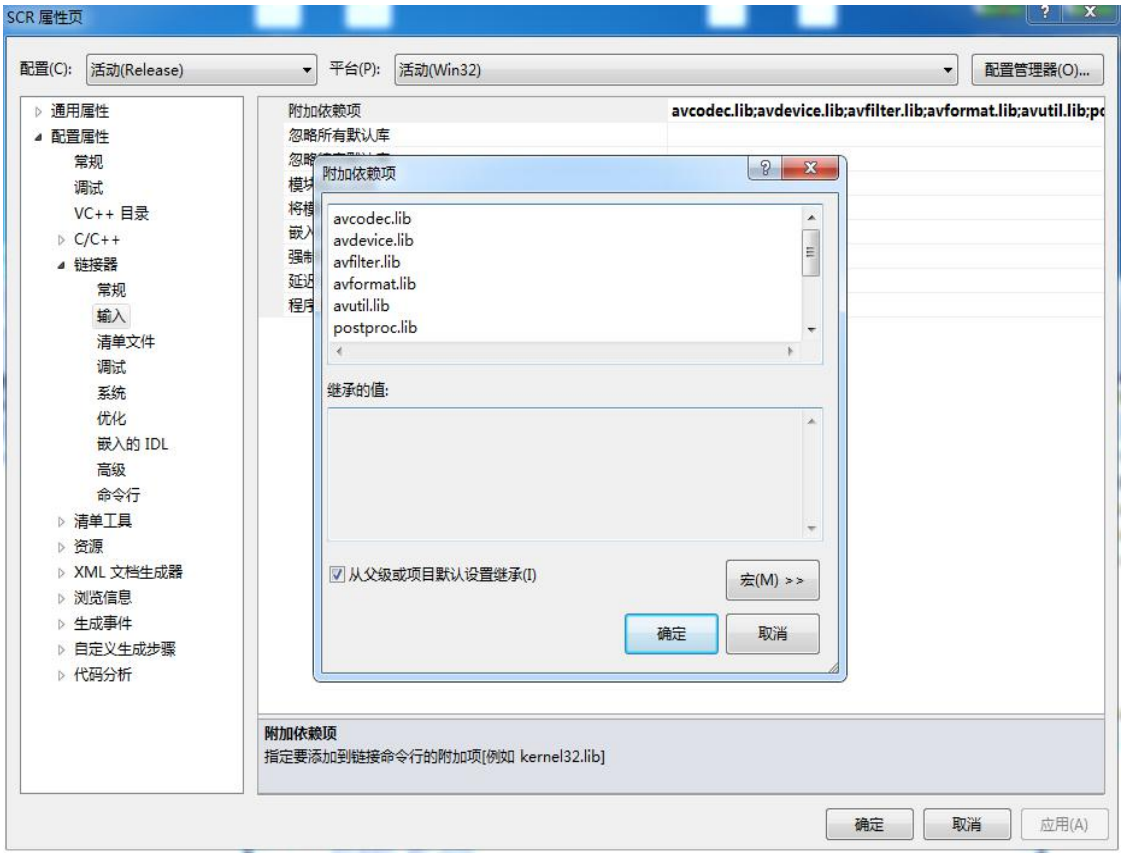


图 4.7　设置附加依赖项

Fig. 4.7　Set additional dependencies

至此，FFmpeg 在 VS2010 下的配置完成。

4.1.2　SDL 在 VS2010 下的快速配置

（1）从 SDL 官网下载与电脑匹配的版本：

（2）环境变量配置

右键单击计算机->属性->高级系统设置->环境变量->系统变量（所有用户都可以使用）或管理员用户变量（只有这个用户可以使用）。在系统变量中拖动滑块框，找到

变量路径，单击 Edit，将路径添加到 bin 文件夹中，注意前面的路径用“;”分隔。如果是 64 位计算机，将其配置进 x64 文件夹，这与 FFmpeg 配置类似。

图 4.8　SDL 下载版本选择

Fig. 4.8　SDL download version selection

（3）VS2010 的配置

方法与 FFmpeg 配置类似。

4.2　基于 FFmpeg 的视频数据采集实现

视频数据采集的基本思想是利用 FFmpeg 的交互式类库 libavdevice，读取计算机的多媒体信号并将数据发送到指定的多媒体设备上。在该模块中，输入设备为计算机屏幕，输出设备为解码模块的入口。通过读取输入设备流的数据帧，将读取的视频数据发送到解码模块以完成视频采集功能。

4.2.1　视频采集设备名获取

在获取主讲人电脑屏幕前，需列出采集设备名。在 Windows 平台下利用 ffmpeg.exe 在 cmd 下读取 DirectShow 设备数据。

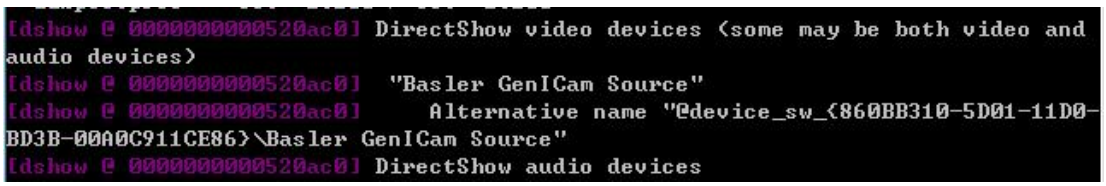


图 4.9　采集设备名

Fig. 4.9　Acquisition device name

4.2.2 视频采集速率控制

采样率和帧率是视频属性的重要参数，是实现播放控制、视频编码、网络传输等功能的基本参数。一般来说，较高的采集速率和帧率直接反映了视频视觉感知的质量。同时，为了追求更高的刷新率和采用率，对硬件的处理能力、编码器性能、软件结构等诸多条件提出了更高的要求。以一个画面分辨率为1366×768的笔记本屏幕视频序列为例，50帧每秒的速度，产生的数据每秒是6556800字节，也就是6.5 MB / s，导致CPU、硬盘、内存等硬件设备负载非常高[25]。

因此，在视频应用的各个领域中，根据不同的应用环境，对视频帧率采集率做了明确规定与限制。例如电影的帧率控制在23.976FPS，PAL制电视领域采用25FPS的帧率，而NTSC制电视采用29.97FPS的帧率。在编码中的帧率往往采用分数的形式表示，因此就有了规定中帧率的小数存在[26]。

在本系统中，将采集模块在独立的线程中实现，根据直播系统需求与视频相关基础知识，视频定为25FPS。

视频帧率控制的实现是通过在获取桌面视频流前，通过av_dict_set()函数将framerate 字段设置为25帧。其函数声明如下：

int av_dict_set(AVDictionary **pm, const char *key, const char *value, int flags)

函数第一个参数是保存了各种设置参数的AVDictionary选项指针。例如framerate、video_size、pixel_format等。第二个参数表示所要设置的参数名称，例如framerate、video_size等，第三个参数表示对应参数的设置值。最后一个参数为标记位，通常设置为空[27]。

此外，在对于桌面视频流分辨率获取的实现上，可以不做设置，这样可以对不同分辨率下的桌面进行动态调整，通过分析视频流信息可以得到分辨率参数。

4.2.3 视频数据采集

本系统的屏幕信息采集实现如下：首先，在SCR动态链接库中封装屏幕信息的采集和过滤方法，在客户端软件的安装期间，SCR会在操作系统中自行注册，然后系统中的API接口调用封装好的DLL来获取屏幕信息。如果采集到视频流，则进一步确定视频帧的位置，并根据捕获的视频帧搜索并打开相应的解码器。最后，循环读取捕获的视频流，并将视频帧保存到AVPacket结构中。如果视频帧被成功读取，则解码器会进行解码，若失败则返回循环。如图4.10所示。

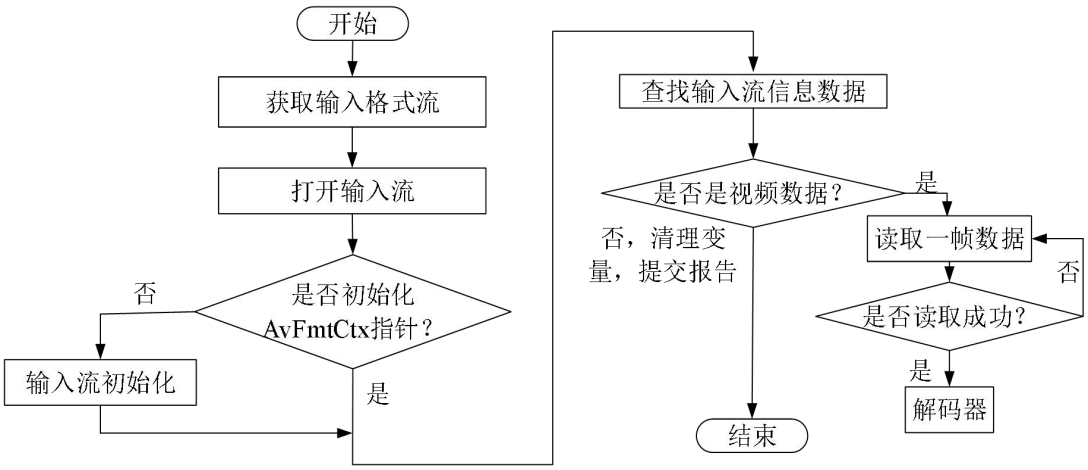


图 4.10　视频画面采集流程

Fig. 4.10　Video screen capture process

4.3　视频编解码具体实现

将视频解码和编码分为两个独立的线程同时工作，使帧速率达到 HD 标准，提高了编码和解码过程的稳定性。解码数据通过 AVFifoBuffer 结构高速缓存指针写入视频缓存块。在编码线程中，通过循环查询缓存区域，读取视频数据帧进行编码。

4.3.1　视频解码器框架

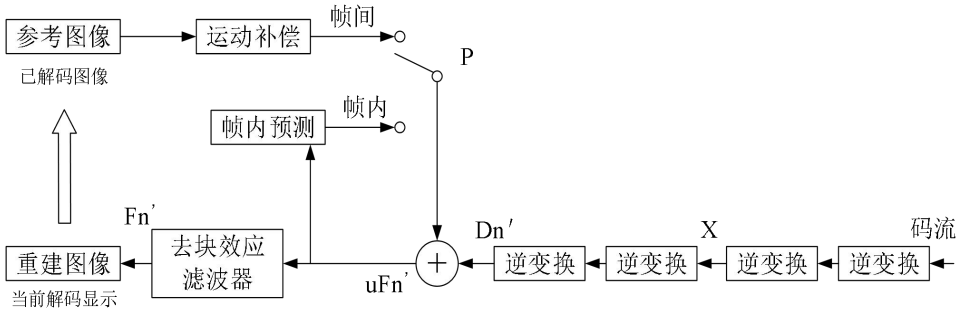


图 4.11　解码原理框架

Fig. 4.11　Decoding framework

解码器的流程框架与编码器形成互补结构，当编码码流进入解码器时，解码器通过利用熵解码对接收到的压缩数据进行重新排序来获得宏块的剩余系数和量化系数 X，之后，解码器进行逆量化和逆变换得到残差宏块 Dn'，之后根据比特流中的头信息进行重

构预测块 P，当预测块 P 与残差宏块 Dn' 相加后得到 uFn'，通过去块效应滤波便得到最后解码输出的图像 Fn'。如图 4.11 所示。

4.3.2 视频解码

流媒体数据是由 RTMP 协议封装的一组视频数据。流媒体数据的基本单元是流，视频数据的基本单元是帧。当客户端接收到流媒体数据时，首先根据传输协议对其进行解析，然后解码器对视频文件进行解析，得到视频的原始帧数据。然后可以对原始数据进行相应的处理来显示图像。视频解码功能的实现需要通过 avformat_find_stream_info()函数获取部分视频流数据，并从中获得相关的视频信息。通过对每个媒体流的 AVStream 结构赋值，实现了解码器的初始查找、打开和视频帧读取功能。

系统获得视频对应的帧信息后，应用解码帧缓存，设为 25 帧，对视频帧大小的缓存进行解码，并设置缓存块的像素空间格式、帧大小、分辨率等参数。最后，使用记录帧位置来确定相应的解码器 ID，并对其进行初始化，对解码结构体缓存设置初始值，申请内存，然后打开解码器，读取视频数据，开始循环解码。

视频解码前，读取到的视频流数据保存在 AVPacket 结构体内，结构体数据成员包含显示时间戳、解码时间戳、视频标识等信息。解码后，将解码出的 YUV 数据存储在 AVFrame 结构体内。AVFrame 包含了较多的解码信息与结构成员变量，如 data[]中存储打包的数据或平面信息，可以直接用 fwrite 函数将 data 中的 YUV 数据写到本地文件内。而 pict_type 标识图像的帧类型，如关键帧 I、预测帧 P 等等。sample_aspect_ratio 表示视频图像的宽高比，以分数形式呈现。

AVFrame 中的 QP 表存储了内存中视频图像宏块的 QP 值，其标号按行由左向右开始。其中 qscale_table[1]的 1 代表第一行中第二列图像宏块的 QP 值，而 qscale_table[2]则代表第一行中第三列图像宏块的 QP 值[28]，因为宏块的大小通常是 16×16 的，因此对于每行宏块数的计算公式为：

$$m_{stride} = width / 16 + 1 \tag{4.1}$$

而宏块的总数为：

$$m_{sum} = ((height + 15) >> 4) \times width / 16 + 1 \tag{4.2}$$

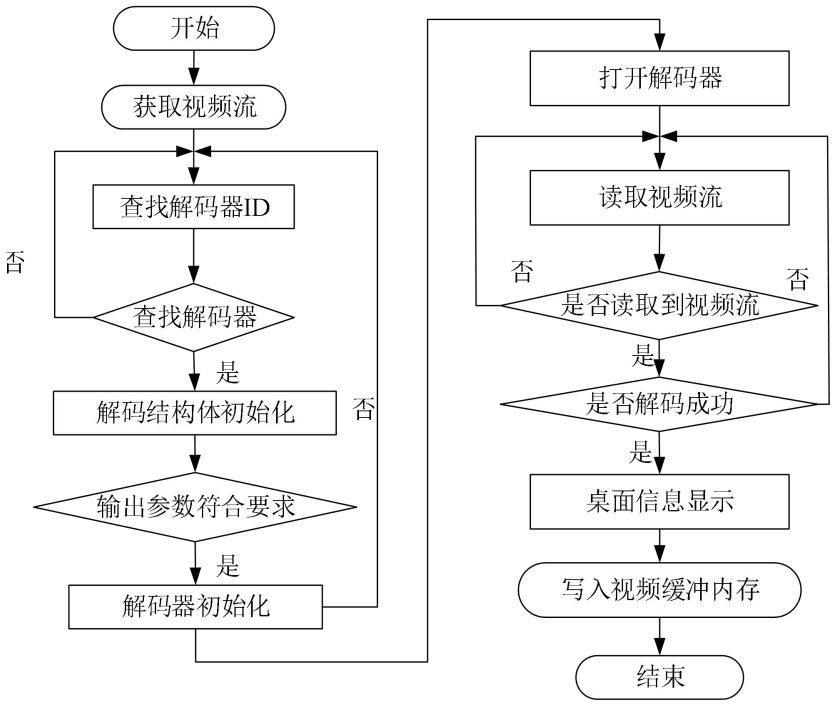


图 4.12　视频解码流程图

Fig. 4.12　Video decoding flow chart

4.3.3　视频播放

（1）流媒体图像显示

SDL (Simple DirectMedia Layer)是基于 C 语言的跨平台开源多媒体开发库[29]。SDL 为用户提供了各种输入和输出功能来对图像、声音进行控制，其跨平台的设计使得开发者可以轻松移植和开发不同平台上的应用程序。目前，SDL 主要应用于媒体播放、游戏等领域。在本文中，使用 SDL 框架渲染、纹理处理解码后的数据以实现高性能的屏幕显示。SDL 显示解码数据的过程如图 4.13 所示。整个 SDL 显示过程可以描述为：

解码后的数据流从缓冲区读入 SDL 处理器。在用标志位检测到视频数据的第一帧后，检测并调用 screen=SDL_SetVideoMode(screen_w,screen_h,0,SDL_FULLSCREEN)来生成视频窗口播放 YUV 格式数据流，并同时设置窗口的相关参数。SDL 创建刷新线程 sfp_refresh_thread 来控制视频数据的循环读取并触发 SDL 窗口刷新事件；刷新窗口事件需要首先计算视频显示区域面积，再由 SDL 渲染器调用 SDL_DisplayYUVOverlay()函数解码视频数据，并作为输入画在显示窗口上，以减少渲染的负担。这样便提高了帧率，从而得到更清晰、更高质量的图像。

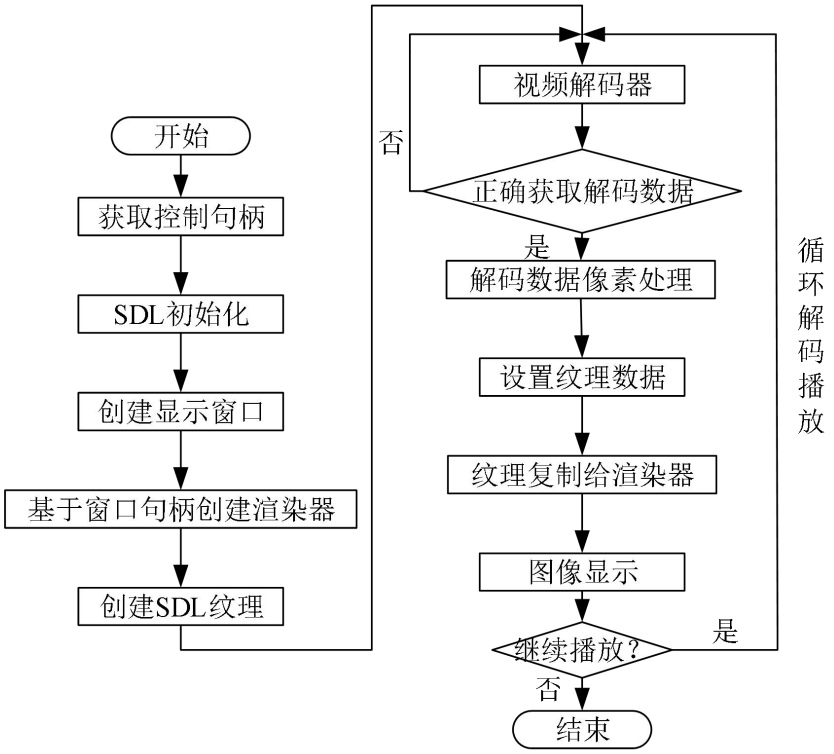


图 4.13　SDL 显示流程

Fig. 4.13　SDL display Flow

（2）实时播放程序优化

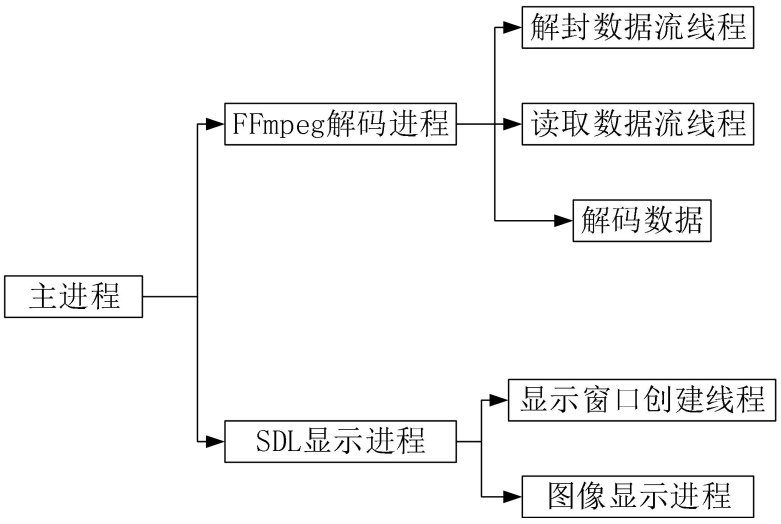


图 4.14　实时播放优化

Fig. 4.14　Real-time playback optimization

在流媒体数据的实时播放过程中，FFmpeg 对流数据进行解析，读取、解码流数据，SDL 生成播放窗口，动态刷新窗口数据将占用大量系统资源。这就需要对整个播放程序进行优化，因此使用 FFmpeg 解码进程和 SDL 显示进程并行操作，这样不仅可以提高系统稳定性，还可以节省系统资源。如图 4.14 所示。在 FFmpeg 解码过程中，创建解析数据流、读取数据和解码数据的线程。在 SDL 显示进程中，创建生成播放窗口的线程和显示图像线程。两进程同时操作且共享资源区域空间不仅节省了系统资源的空间，还减少了许多系统中的复杂操作，使得系统效率得到大幅提升。

4.4 本章小结

本章主要完成了客户端软件视频处理技术的具体实现。首先对 FFmpeg 编译移植和软件开发环境的配置进行阐述说明，接着对直播软件中视频数据采集、解码、编码等关键技术进行实现，并且在对实现过程中的新技术进行详细阐述，包括对视频数据采集速率控制、单独线程的编解码实现等。

5 基于 Nginx 服务器的搭建与数据传输实现

Nginx 是一个免费的开源 web 服务器软件，由俄罗斯软件工程师 Igor Sysoev 使用 C 语言从零开始开发。它于 2004 年发布，专注于各种 web 服务器特性：负载均衡、缓存、访问控制、宽带控制[30]。它具有稳定性高、并发连接支持好、内存消耗少、功能丰富等优点，越来越受到世界各国的关注。

5.1 高并发的重要性

随着 web 服务的兴起，并发性已经成为 web 和服务器体系结构面临的最大挑战[31]。在早期，并发问题的主要原因是客户机的访问速度慢。然而，如今大量的多媒体数据或图片数据消耗服务器资源，大型网站通常采用分离图片或多媒体服务器的方法来避免这种风险。然而，从移动和新的应用程序体系结构来看，web 服务器体系结构通常对诸如持久连接更新之类的服务要求更高。

对于 Apache 体系结构（它目前拥有互联网的最大份额），网站生成小于 100KB 的响应，而页面的生成可能最多需要 1 秒钟，并且在连接之前将整个页面发送到客户机最多需要 10 秒钟可以关闭。结果，随着持久连接应用的并发处理问题日益严重，为了处理不断增加的用户负载和高阶量的并发任务，除了提升服务器组件效率外，还提高了网络带宽等基础设施，Web 服务器的性能也应该能够支持扩展的升级。

在处理并发问题时，Nginx 没有为服务请求建立新的过程，而是基于事件模型。由于并发连接处理，传统的基于进程或线程的模型可以阻塞网络或 I/O 操作，不仅如此，包括内存分配和初始化，还需要额外的 CPU 时间。因此，Nginx 使用大量的多路复用和事件通知，并根据类型将不同的任务分配给不同的进程，并使用高效的单线程循环处理连接，从而可以在普通硬件上处理较高的并发连接。

5.2 Nginx 架构分析

Nginx 模块化架构允许开发人员在不修改核心的情况下自由扩展 Web 服务器的功能。它可以分为核心模块、事件模块、协议模块(HTTP)、负载均衡器等。每个模块代码与 Nginx 的核心代码一起编码。此外，核心模块还封装了内存池、网络连接处理、自旋锁等操作的实现，相应的源代码实现分别在 ngx_spinlock.c 、 ngx_palloc.c 、 ngx_connection.c 下[32]。图 5.1 展示了 Nginx 架构设计。

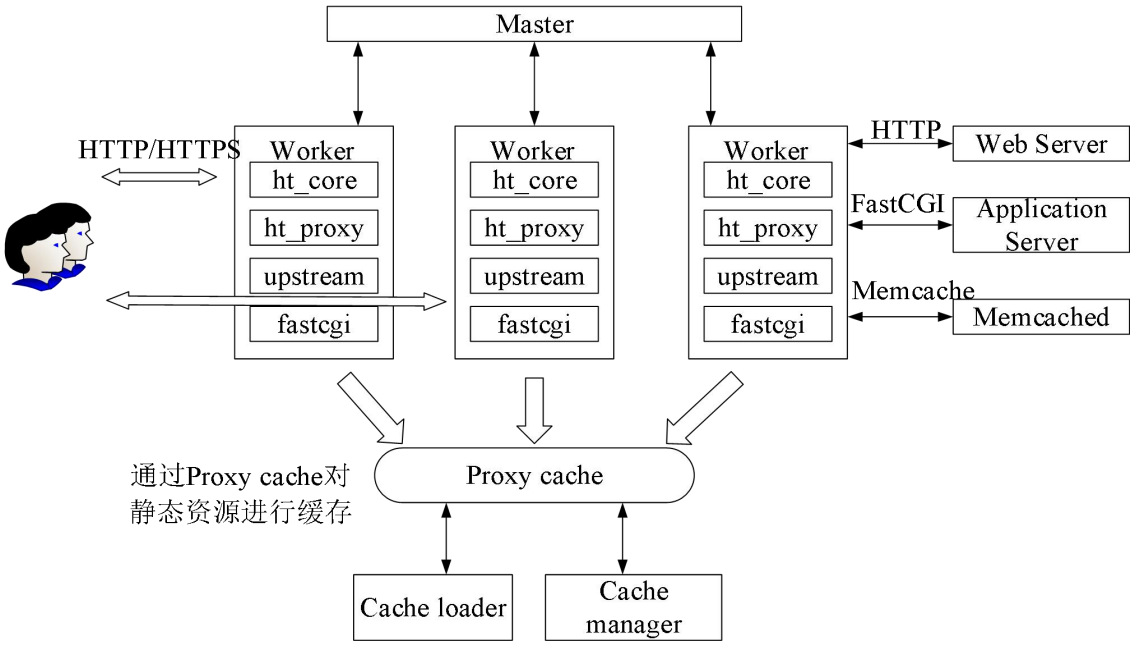


图 5.1　Nginx 高层架构设计

Fig. 5.1　Nginx high-rise architecture design

通过分析 Nginx 的架构，可分析出监听套接字在 Nginx 初始化启动阶段便已经完成，启动程序将包括一个 Master 进程和多个 Worker 进程，Master 负责外部信号或接收请求，通过 Worker 进程使用套接字来接收和读取请求，并作出输出响应，通过高效的事件处理周期来完成连接处理。Nginx 还支持 HTTP 下的 SSL 安全协议。

5.3　Nginx 配置文件

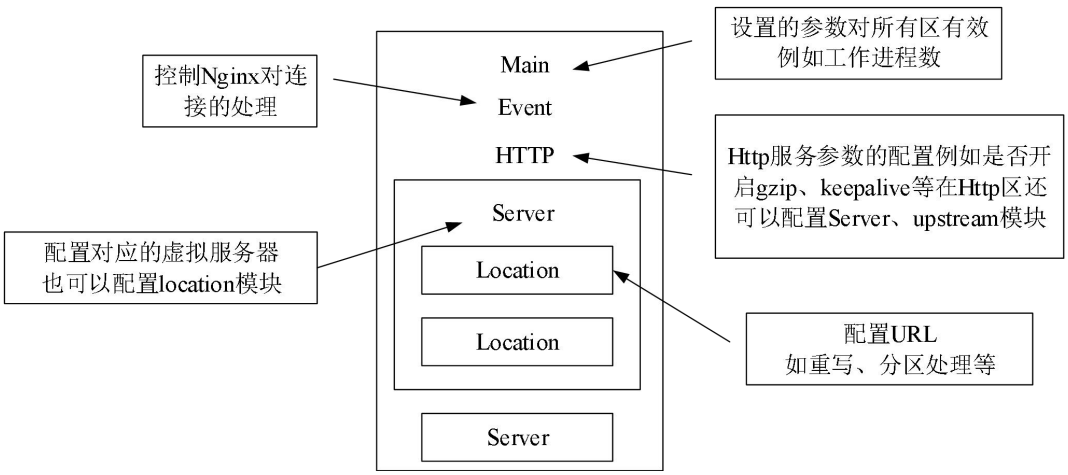


图 5.2　Nginx 配置文件分区说明图

Fig. 5.2　Nginx profile partition diagram

配置文件是可扩展性 web 服务器的重要组成部分。对服务器的日常维护一般会消耗大量的人力和物力。而 Nginx 重新设计的 C 风格配置文件正好解决了这一问题。

Nginx 的配置文件通常是名为 nginx.conf 的文本文件，它不仅支持正则表达式，还按区域划分为全局区域、事件区域、HTTP 区域、服务器区域、位置区域等。每个配置区域的描述如图 5.2 所示。

5.4 基于 Nginx 的 RTMP 配置文件

Nginx 本身是一个具有高稳定性、高并发连接支持、低内存消耗和丰富功能的 HTTP 服务器，世界上越来越多的人已经注意到它的优点。而 FFmpeg 又是非常优秀的开源视频技术解决方案，因此，本课题结合 FFmpeg，采用基于 RTMP 的 Nginx 服务器作为智能录屏及播放系统的流媒体服务器。

（1）下载 Nginx 所需要的模块安装包并安装，所需文件如图 5.3 所示：

```
[root@centos-rpi3 nginx]# ll
total 1436
drwxr-xr-x. 9 1001 1001    4096 Jan  1  1970 nginx-1.10.0
-rw-r--r--. 1 root root  908954 Sep  7  2016 nginx-1.10.0.tar.gz
-rwxr-xr-x. 1 root root    2725 Jan  1  1970 nginx-install.sh
drwxr-xr-x. 6 root root    4096 Aug  9  2016 nginx-rtmp-module-master
-rw-r--r--. 1 root root  545752 Sep  7  2016 nginx-rtmp-module-master.zip
```

图 5.3　模块安装包

Fig. 5.3　Module installation package

（2）配置并编译 Nginx

使用 Nginx 提供的默认服务器配置并添加 RTMP 模块。

（3）运行测试 Nginx

```
[root@centos-rpi3 sbin]# ./nginx
[root@centos-rpi3 sbin]#
```

图 5.4　运行命令

Fig. 5.4　Run the command

进入安装位置/usr/local/nginx/sbin，运行命令./nginx。

打开浏览器，并在地址栏输入服务器的 IP，如图 5.5 所示表示 Nginx 服务器已经搭建成功。

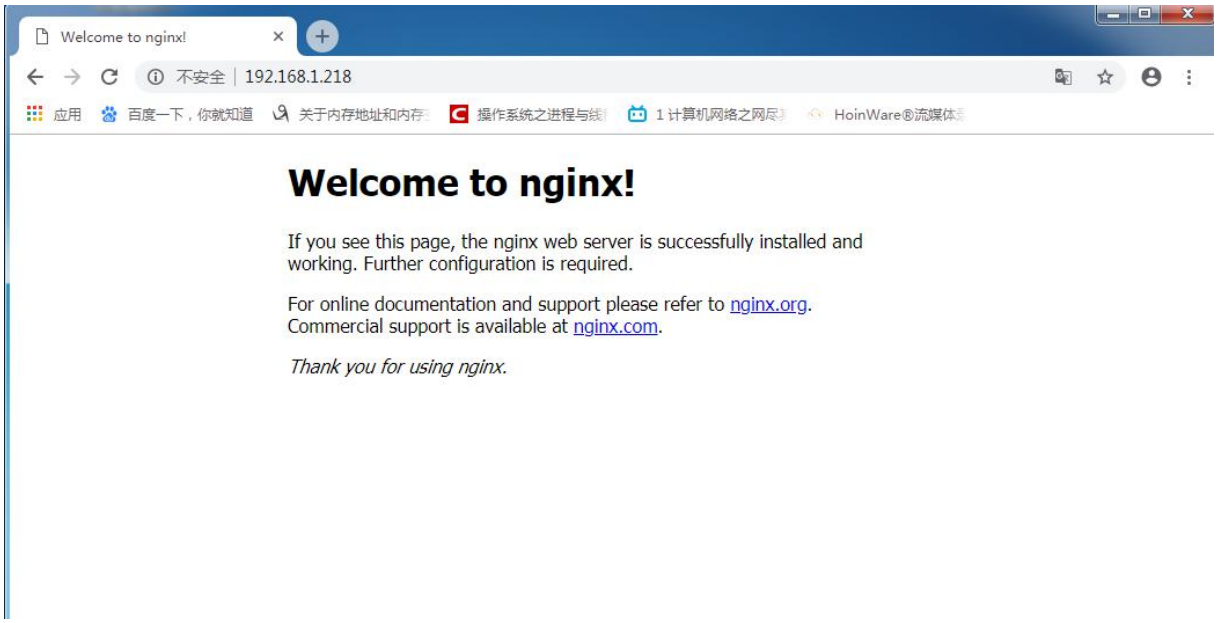


图 5.5　服务器搭建成功

Fig. 5.5　Successful server setup

5.5 流媒体服务器直播配置

首先，设置并发值。在安装 Nginx 后，默认的最大并发为 1024，如果增加并发数量，可以利用 worker_connections 修改。当然，还需要根据实际需求进行配置，当数值设置太高，会增加 cpu 负载，因此根据反向代理模式下最大连接数的理论公式：

$$\text{最大连接数}=worker\_connection \times worker\_processes / 4 \qquad (5.1)$$

对于直播服务，需要在 server 字段添加应用名，这里设置为 live，提供多个频道直播的开启。开启 live、live2 两个频道之后，重新启动 nginx 并打开浏览器，就会看到 live 与 live2 字段，证明配置已经生效，如图 5.6 所示。

RTMP	#clients	Video				Audio				In bytes	Out bytes	In bits/s	Out bits/s	State	Time
Accepted: 2		codec	bits/s	size	fps	codec	bits/s	freq	chan	2.65 MB	0 KB	0 Kb/s	0 Kb/s		19s
live															
live streams	2														
4001482740939	1	H264 High 3.1	0 Kb/s	1280x720	25	AAC LC	0 Kb/s	48000	2	365 KB	0 KB	0 Kb/s	0 Kb/s	active	0s
4001482742932	1	H264 High 3.1	0 Kb/s	1280x720	30	AAC LC	0 Kb/s	44100	2	2.29 MB	0 KB	0 Kb/s	0 Kb/s	active	5s
live2															
live streams	0														

图 5.6　直播服务配置成功

Fig. 5.6　The live service configuration was successful

6 系统测试及分析

初步完成基于 FFmpeg 和 SDL 的智能录屏及播放系统后，还需要对各项功能进行测试与评估。并且通过不断的测试，才能更新并且纠正系统在实现上的错误与不完善的地方，系统内部的某些参数和变量也是在不断修改中达到实际效果最佳的数值。因此，本章将会对直播系统的系统功能、网络延时、视频传输质量等进行系统性的测试。

6.1 系统基本功能测试

本章主要对系统基本功能的实现程度进行了测试并考察系统在系统在正常工作流程中的各项参数能否满足初始设计要求。

6.1.1 系统测试环境

首先测试整体系统的基本功能，目的是检测系统对基本功能的实现效果。本次测试的系统硬件环境如表 6.1 所示：

表 6.1 设备配置

Tab. 6.1 Equipment configuration

类型	系统配置	参数
客户端	操作系统	Windows 7
	CPU	i7-6700
	内存	8G
	硬盘	1T
服务器	操作系统	Linux

从表中可以看出本次测试的硬件环境条件一般，相比于目前大部分计算机，测试机器的各项性能都较低，因此测试结果具有很高的适用性。

6.1.2 安装流程测试

基于 FFmpeg 和 SDL 的智能录屏及播放系统通过 InstallShield Wizard 对软件进行封装发布，包含了软件需要使用的动态链接库以及可执行文件。在打包完成后，对安装项目进行调试运行，生成安装包。和普通软件安装过程类似，双击运行便可以进入安装过程。

在安装过程中会提示用户对安装目录进行选择，最后在指定的目录下，安装程序会将封装文件解压出来，并在桌面生成快捷方式。

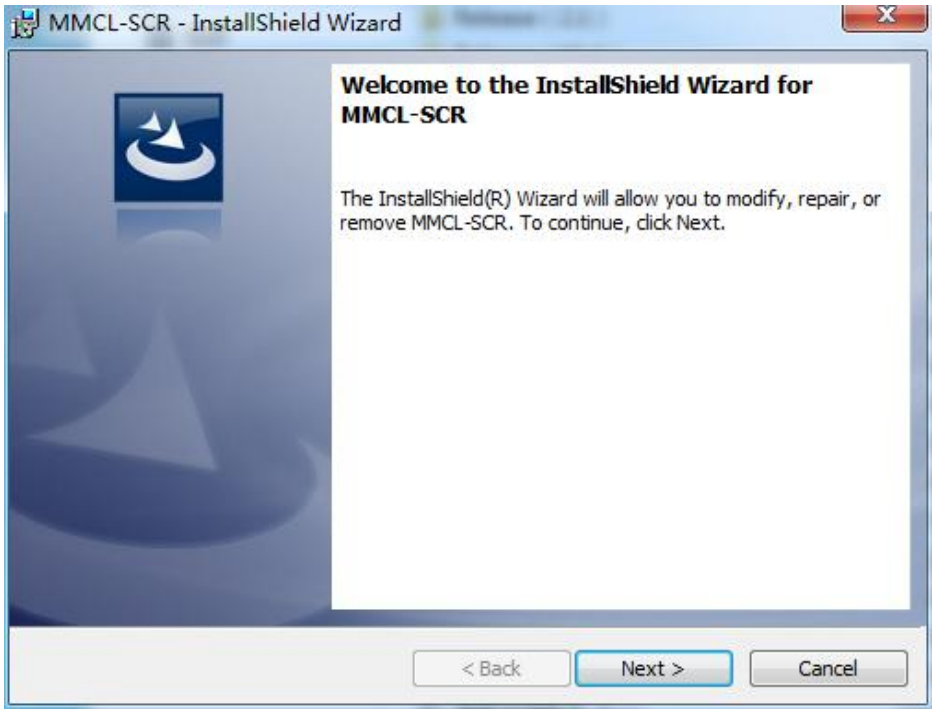


图 6.1　安装界面

Fig. 6.1　The installation interface

6.1.3　基本功能测试

运行客户端软件，单击投影界面上的开始按钮，则自动开始推送屏幕数据。客户端连接到服务器后，服务器便会接收到屏幕信息，经过服务器数据转发以后，接收端可以通过软件接收。

基本功能测试结果如图 6.3 所示。从图中可以看出，系统的基本功能初始的设计要求，系统延迟相对较小，大约 500ms，与传统的多媒体传输方式相比，实时观看会以牺牲图像质量为代价，或提供高清晰度图像，时延约 10s。本系统在保证系统处于低时延工作状态的同时，能够满足一定的高质量图像。低时延对实时会议系统、远程教学、游戏直播具有很强的现实意义。此外，由于系统采用 Raspberry Pi 3B 作为服务器端的硬件平台，具有很强的可移植性且没有时间和地点的约束。

图 6.2　软件界面

Fig. 6.2　Software interface

图 6.3　基本功能测试

Fig. 6.3　Basic functional test

6.2　系统稳定性测试

系统稳定性测试，是通过仿真软件将业务压力定量地施加到需要测试的部分，并确保系统可以在这种业务压力仍可以保证正常工作一种测试方法。

为了测试系统稳定性，本文设计并采用了两种不同的测试实例，分别是单机推送/单机接收和多设备推送/多设备接收。测试方法是在实验室搭建并启动基于 FFmpeg 和 SDL 的智能录屏及播放系统，使系统在不同压力下连续工作 48 小时。

6.2.1 单机推送和单机接收

表 6.2 测试结果

Tab. 6.2 Test result

序号	播放时长	延迟时间	稳定性
1	1h	<500ms	稳定
2	2h	<500ms	稳定
3	3h	<500ms	稳定
4	6h	<500ms	稳定
5	9h	<500ms	稳定
6	12h	<500ms	稳定
7	24h	<500ms	稳定
8	48h	<500ms	稳定

在实验室（大连理工大学创新园大厦 A501，面积 7m×9m）环境良好的情况下，一台设备（CPU 型号 i5-4330，内存 8G）用作推送设备，另一台设备（CPU 型号 i5-3210M，内存 4G）用作接收设备，完成单机推送/单机接收实验。实验结果如表 6.2 所示。

可以看出，系统经过连续 48 小时测试仍能保证稳定工作，工作期间整体系统延时小于 1 秒，能够满足正常使用。

6.2.2 多设备轮流推送测试

表 6.3 配置清单

Tab. 6.3 Configure list

编号	CPU	内存	硬盘	备注
1	i5-4330	8G	500G	推送/接收设备
2	i7-5500U	8G	1T	
3	i5-3210M	4G	500G	
4	Z2760	2G	64G	

在实验室（大连理工大学创新园大厦 A501，面积 7m×9m）环境良好的情况下，一台设备用作推送设备，其他三台设备用作接收设备，每隔一段时间切换推送端来测试系统切换速度，系统响应及系统稳定性。多设备轮流推送实验设备配置清单如表 6.3 所示，实验结果如表 6.4 所示。

表 6.4　测试结果

Tab. 6.4　Test result

序号	播放时长	延迟时间	稳定性
1	1h	<500ms	稳定
2	2h	<500ms	稳定
3	3h	<500ms	稳定
4	6h	<500ms	稳定
5	9h	<500ms	稳定
6	12h	<500ms	稳定
7	24h	<500ms	稳定
8	48h	<500ms	稳定

可以看出，系统经过连续 48 小时测试仍可以稳定工作。实验期间，设备切换功能正常，整体延迟小于 1 s，能够满足正常使用。

6.2.3　系统延时测试

系统延迟是指完成系统对数据接收传输准备所需要等待的时间，这个时间越短，表明系统性能越高，用户体验也就越好。为了测试系统延迟，本文采用三组不同区域、不同用户数量、不同并发量的对比实验进行测试，。

根据测试区域，将系统延迟试验分为三组。第一组测试实验室环境，第二组测试大连理工大学研究生教学大楼中的标准小教室，第三组测试大连理工大学机械馆阶梯教室，对比试验所使用的直播平台为虎牙直播。

（1）实验室环境测试

本实验在实验室（大连理工大学创新园大厦 A501，面积：7m×9m）环境良好的情况下，由一台设备（CPU 型号 i5-4330，内存 8G）用作推送设备，另一台设备（CPU 型号 i5-3210M，内存 4G）用作接收设备。反复切换 PPT 页面，记录每次切换时的延迟。

在教研室内，本系统测试结果如图 6.4 所示。由折线图可以看出，总延时可以保证在 1s 以内，能够满足正常使用。

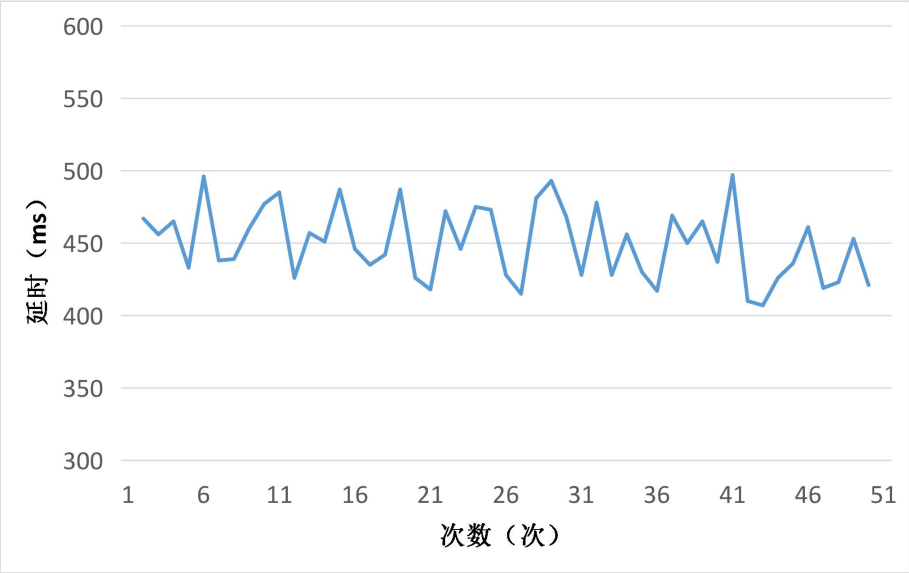


图 6.4　智能录屏及播放系统延迟测试结果

Fig. 6.4　Delay test results of intelligent video recording and playback system

在教研室内，虎牙直播延时测试结果如图 6.5 所示。由折线图可以看出，总延时在 9 秒左右，比本系统延时高出一个数量级，这是实时直播完全不可忍受的，不满足实时直播需求。

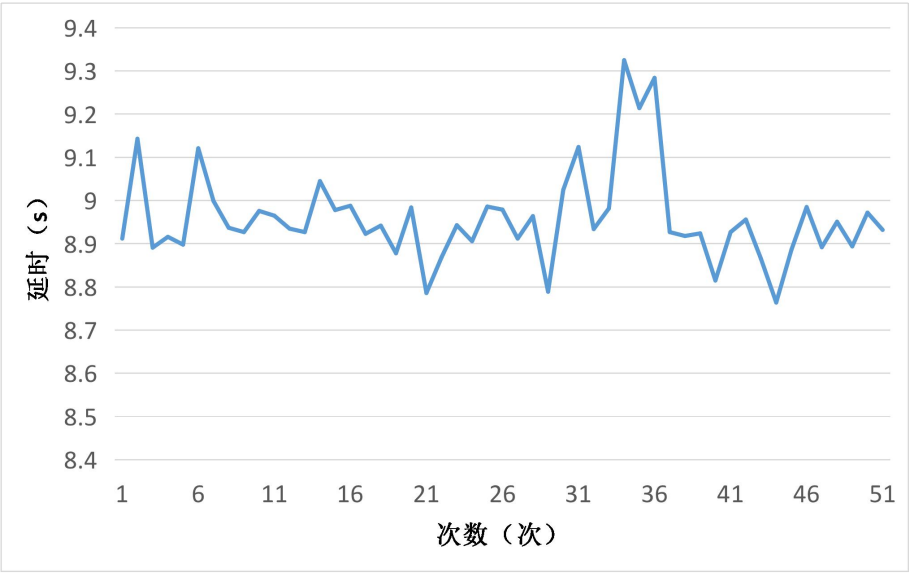


图 6.5　虎牙直播延迟测试结果

Fig. 6.5　Tiger live broadcast delay test results

（2）标准小型教室环境测试

标准小型教室环境测试实验场地为大连理工大学研究生教育大楼 405 教室（面积：6m×8m），实验设备部署于实验区最前方。实验环境如图 6.5 所示，本次实验主要测量的是在教室教学情境下，智能录屏及播放系统的稳定程度和时延程度。实验设备配置列表及实验结果如表 6.5 和 6.6 所示。

表 6.5　智能录屏及播放系统测试结果

Tab. 6.5　Intelligent video screen and playback system test results

编号	CPU	内存	硬盘	延时
1	AMD A4-3305M	4G	500G	400ms
2	AMD A8-4500M	4G	500G	800ms
3	i3-3120M	4G	500G	570ms
4	i5-4200U	4G	500G	390ms
5	i5-3230M	4G	500G	300ms
6	i5-3210M	4G	500G	130ms
7	Pentium T4500	2G	400G	320ms
8	i7-3632QM	16G	1T	500ms
9	i7-5500U	8G	1T	400ms
10	Z2760	2G	64G	600ms

表 6.6 虎牙直播测试结果

Tab. 6.6　Tiger live test results

编号	CPU	内存	硬盘	延时
1	AMD A4-3305M	4G	500G	8.780s
2	AMD A8-4500M	4G	500G	9.480s
3	i3-3120M	4G	500G	9.140s
4	i5-4200U	4G	500G	8.920s
5	i5-3230M	4G	500G	8.820s
6	i5-3210M	4G	500G	8.725s
7	Pentium T4500	2G	400G	8.970s
8	i7-3632QM	16G	1T	9.020s
9	i7-5500U	8G	1T	8.790s
10	Z2760	2G	64G	9.310s

结果表明，在标准小型教室（面积：6m×8m）环境下，智能录屏及播放系统系统可以满足正常使用需求。实验期间，系统客户端软件可以完美地在实验设备上运行，服务器稳定性能表现良好。而虎牙直播由于其高延时，学生接收到的画面与授课 PPT 严重脱节，严重影响课堂授课质量，不满足使用需求。

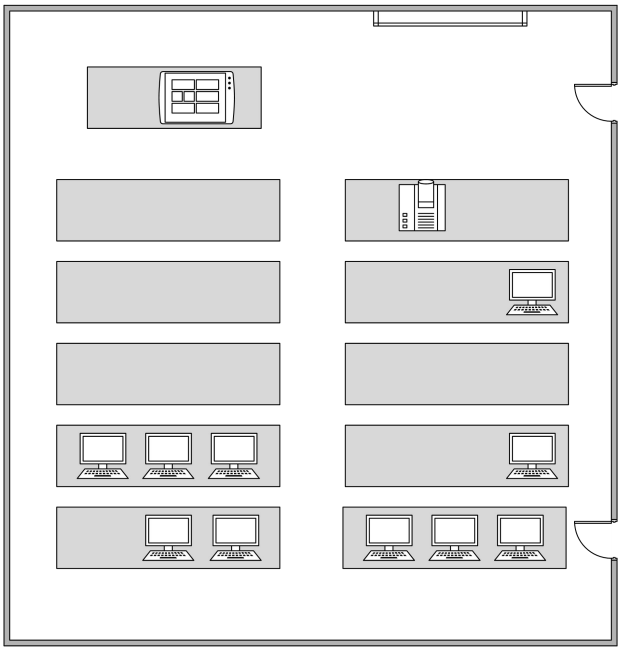


图 6. 6　小型教室环境测试

Fig. 6.6　Classroom environment test

（3）大教室环境测试

在学校或大型展览中，经常会遇到各种各样的应用场景，对于那些面积大，环境复杂的场景来说，信息共享是困难的，比如在大型课堂教学过程中，学生由于投影范围有限，无法在课堂后面看到屏幕上的内容，很难和老师进行交流。因此，接下来在阶梯大教室中进行实验，以测试系统在较大空间的环境，能否为用户提供稳定的服务。

实验地点为大连理工大学机械楼 251 教室（面积：12m×14m）。实验设备部署于实验区最前方，讲师计算机和所有设备通过局域网进行联通。讲师设备将屏幕信息推送到服务器，用户使用客户端软件进行接收。实验环境的基本数据和每个设备的放置如图 6.7 所示。本实验在实际上课的情境下，对系统的稳定性和延时进行了现场测试。

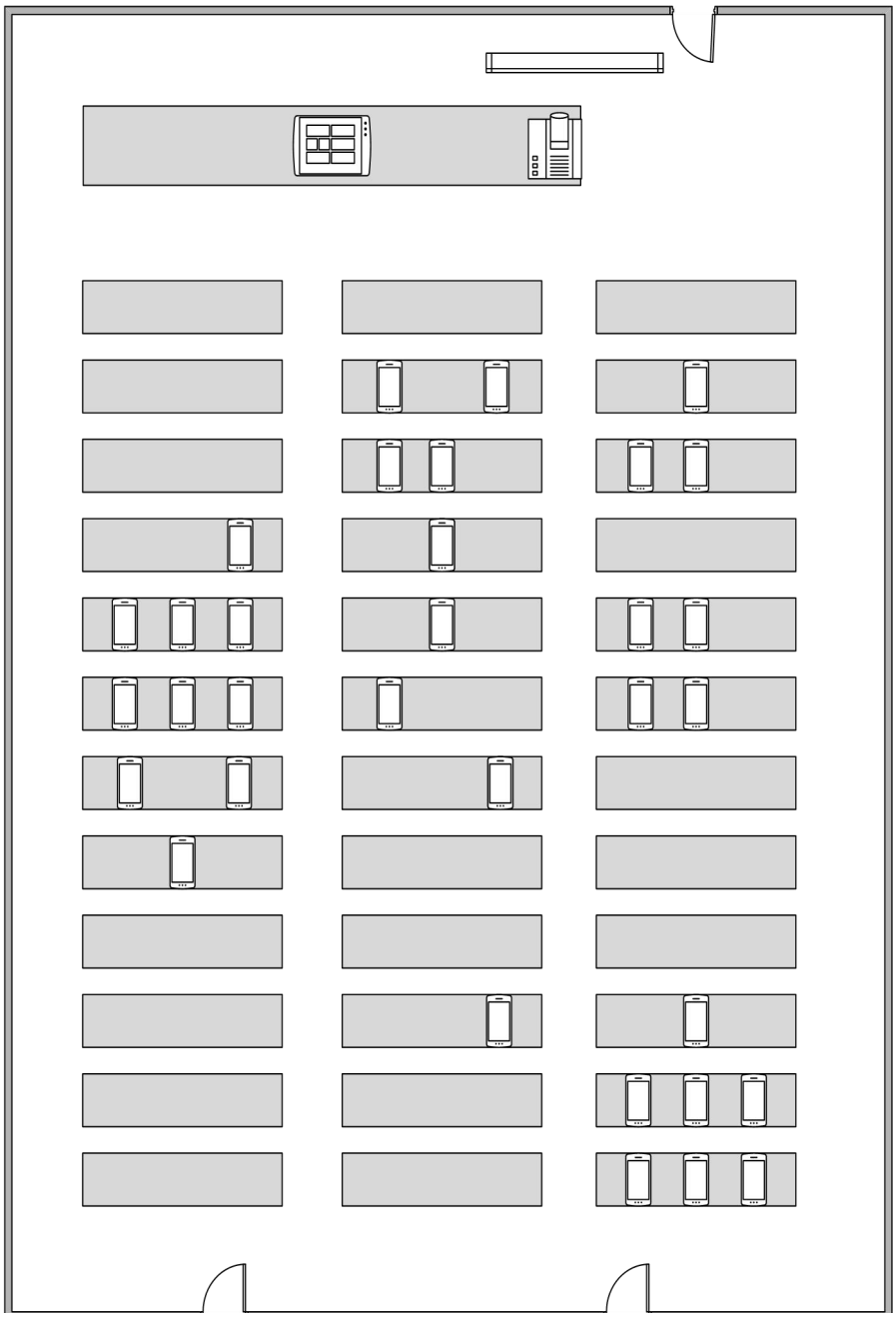


图 6.7　大型教室环境测试

Fig. 6.7　Classroom environment test

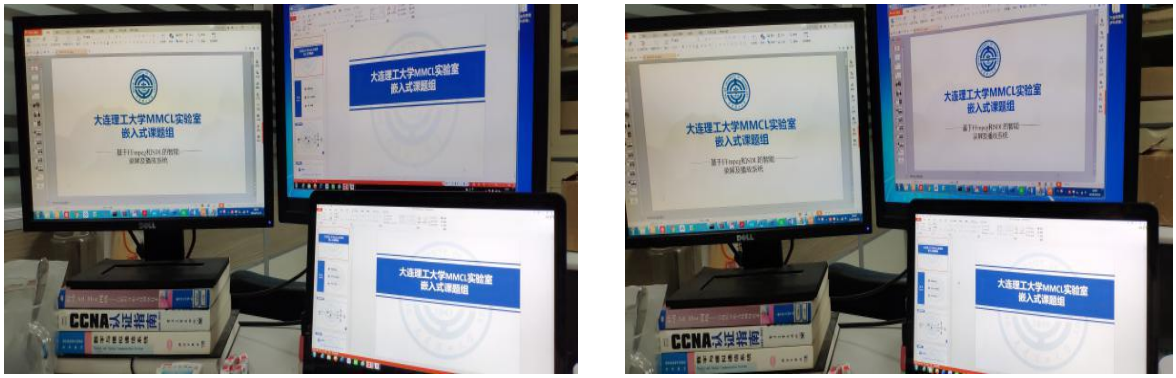
实验期间，学生可以通过随身携带的设备接收老师的 PPT 画面与教师保持同步，系统延迟稳定在 1s 以内，能够满足正常使用需求。与标准小型教室实验相似，虎牙直播由于高延时不能够满足正常使用需求。

6.2.4 用户切换测试

用户切换测试是在系统正常工作时通过切换推送设备来测试切换速度和系统稳定性。用户切换时间越短则表明系统性能越好。

实验场地为大连理工大学创新园大厦 A501 创新园区（面积 7m×9m）。选择其中一台设备用作推送设备来提供屏幕数据信息，使用另一台设备作为接收来测试系统的工作状态。

通过现场实验，在图 6.7 中展示出了实验效果。得出以下结论：多个设备执行切换操作，系统能够对所有设备做出正常响应，实现界面平滑切换，能够满足正常使用需求。



（a） （b）

图 6.8 切换用户测试

Fig. 6.8 Switch user test

6.2.5 系统兼容性测试

系统兼容性测试是指在不同型号和版本的不同软硬件平台上运行客户端软件，测试其工作稳定性。本实验主要是针对 Windows 的不同版本和不同的智能硬件，包含以下几组测试样例：Windows7、Windows8、Windows10。测试结果如图 6.8 所示，其中，图（a）是在 Windows7 上运行，图（b）是在 Windows8 上运行，图（c）是在 Windows10 上运行，图（d）是在以 Windows10 为操作系统的平板上运行。

由结果可以看出，本文设计的系统可以满足市面上常用 Windows 智能设备的基本需求。

（a）　（b）

（c）　（d）

图 6.9　系统兼容性测试

Fig. 6.9　System compatibility test

6.3　系统质量评价

在完成了基于 FFmpeg 和 SDL 的智能录屏及播放系统的基本功能测试后，本文从流媒体数据压缩编码损耗和传输损耗两方面对智能录屏及播放系统进行了质量评估，并进行了相关计算和分析。

（1）峰值信噪比(PSNR)

PSNR(Peak Signal to Noise Ratio)峰值信噪比是图像质量的一种评价标准，它把原始参考模型与失真的视频序列逐帧逐像素的进行比较，通过二者之间的误差来判断失真视频和原始视频的相似程度[33]。一般来说，PSNR 值范围在 20-40 之间，PSNR 值越大，处理后的视频失真程度越小。

对于本系统来说，启动客户端软件之后，可截取并保存视频片段至本地，再从接收端截取并保存同时间的视频片段至本地，最后运用公式 6.1 和 6.2 计算两个视频像素点的 PSNR 值。

$$MSE = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} (X(i,j) - Y(i,j))^2 \quad (6.1)$$

$$PSNR = 10 \lg \frac{(2^n - 1)^2}{MSE} \quad (6.2)$$

其中，MSE 为当前图像 X 与参考图像 Y 的均方误差，H、W 分别表示图像的高度和宽度；n 为每像素的比特数，一般取 8，即像素灰阶数为 256。

通过计算视频图像像素，测量序列的平均 PSNR 值为 28。相比于 PSNR 标准，本系统传输过程中视频数据的损耗和失真较小，能够满足正常使用需求。

（2）信号强度

为了验证系统在某些特定环境传输的可靠性，本文对系统覆盖区域内的信号强度进行了测试，并定量地计算了被测环境中某一点的信号强度。以判断该点的信号强度能否满足要求。在一定范围内，数据传输的信号强度应高于-60dB。

$$L_{bf} = 32.45 + 20 \lg f(MHz) + 20 \lg d(km) \quad (6.3)$$

$$f = 2.4G = 2400MHz \quad (6.4)$$

自由空间中的基本传输损耗是由两个理想点源天线在自由空间中的传输损耗来定义的[34]。它代表了自由空间中两个理想点源之间的发射功率和接收功率的比值。计算公式为公式 6.3，本文发射功率取值为公式 6.4。

表 6.7　不同距离的信号强度结果对比

Tab. 6.7　Comparison of signal strength results at different distances

项目	数据结果		
$d(m)$	5.0	10.0	20.0
$f(GHz)$	2.4	2.4	2.4
理论值(dB)	-34.05	-42.40	-46.25
实测值(dB)	-38	-44	-48

对于实际计算，可以通过自由空间中的基本传输损耗来定量分析某一点的信号强度：若无线发射装置的功率为 100mw，发射功率为 20dbm，则距离 10m 所接收到的信

号强度即为 20- L_{bf} 。在实验室环境下，分别对距离为 5m、10m、20m 的不同空间，使用公式 6.2 计算得出不同距离下的 L_{bf} ，再使用信号强度检测工具测出实际信号强度。

表 6.7 表明，公式 6.3 计算得出的信号强度与实测信号强度数据基本一致，在测试范围内，结果均大于-60dB，表明系统可以正常为用户提供稳定的数据支持。

结　论

网络直播技术的发展，不仅丰富了大众的文化生活，而且在某些特殊场合也发挥着重要作用。然而，随着直播系统的大规模应用，在实施过程中也出现了一些不足。因此，为了克服现有直播系统存在的一些问题，本文设计并实现了一种基于 FFmpeg 和 SDL 的智能录屏及播放系统，相较传统的直播系统，本系统在一定程度上进行了优化，具有一定的经济和社会意义。本文主要完成了以下工作：

（1）对基于 FFmpeg 和 SDL 的智能录屏及播放系统实现过程中所采用的关键技术做了研究概述，包括视频颜色空间、高清 H.264 视频编码标准、FFmpeg 视频处理技术与 RTMP 流媒体传输协议，之后提出了基于 FFmpeg 和 SDL 的智能录屏及播放系统系统总体设计方案。

（2）完成对 FFmpeg 源码的移植编译、裁剪与 VS2010 开发环境下的快速配置工作，同时详细研究了 FFmpeg 对视频的处理逻辑流程，并根据系统的需求对 FFmpeg 处理流程进行优化改造。另外，使用 H.264 高清视频编码对视频数据进行高质量压缩编码，不仅提升了视频 PSNR 值且大大减少了网络带宽压力。

（3）完成流媒体服务器的搭建。以 Raspberry Pi 3B 为硬件平台，在 SD 卡中烧写 Centos 操作系统，采用带有 RTMP 模块的 Nginx 服务器来实现视频流的传输，大大降低了数据传输延迟。

（4）根据不同环境下的实地实验结果，对系统整体进行了质量评估。基于 FFmpeg 和 SDL 的智能录屏及播放系统符合设计要求，可以满足不同用户在多种环境下的正常使用。

本文的工作虽然在网络延时、视频处理效率、画面质量等方面取得了一些成果，但考虑到智能录屏及播放系统系统的复杂程度，本系统还可以从以下几个方面进行进一步的研究：

（1）需要扩展流媒体服务器的功能，如增加云数据共享空间、根据用户需求构建不同的分组等功能。

（2）在网络与信息安全日益严峻的今天，需要对直播数据进行加密处理，同时应对系统进行多项安全防护与漏洞排查工作。

（3）本系统只完成了 Windows 平台下的实现，然而，在移动直播领域，仍有必要进行移植和改进系统，比如在 4G 和未来即将出现的 5G 网络环境中动态降低实时流媒体的帧速率和图像质量。

参 考 文 献

[1] 黄诗文.基于 FFmpeg 的高性能高清流媒体播放器软件设计[D].杭州:浙江大学,2012.
[2] 中国互联网络信息中心.中国互联网络发展状况统计报告[R].2018.
[3] 张国庆.基于 FFmpeg 的视频转码与保护系统的设计与实现[D].武汉:华中师范大学,2011.
[4] 孟娇.基于 SSH 框架的农业服务系统的设计与实现[D].西安:电子科技大学,2015.
[5] 吴迪.基于 Nginx 的安全管理系统的设计与实现[D].北京:北京邮电大学,2013.
[6] PRAKASH P, BIJU R, KAMATH M. Performance analysis of process driven and event driven web servers[C]//Intelligent Systems and Control (ISCO), 2015 IEEE 9th International Conference on. IEEE, 2015.
[7] 林丽丽.使用高性能 Web 服务器 Nginx 实现开源负载均衡[J].大众科技，2010,07:121-124.
[8] 田纯青.利用 Nginx 实现基于 URL 的 web 负载分配[J].现代计算机,2009,07:78-82.
[9] 段学静.基于 RFID 技术的高校图书馆管理系统设计与实现[D].沈阳:东北大学,2015.
[10]李翊翔.在线音乐直播的商业模式探究[D].厦门:厦门大学,2017.
[11]VASEK M, WADLEIGH J, MOORE T. Hacking is not random: a case-control study of webserver-compromise risk[J]. IEEE Transactions on Dependable & Secure Computing,2015:1-1.
[12]YUAN W, SUN H, WANG X, et al. Towards efficient deployment of cloud applications through dynamic reverse proxy optimization[J]. IEEE,2013:651-658.
[13]XI-YUAN X U, YAO Y K. The predictive value of APACHEⅡ and TISS-28 in the treatment to respiratory failure in acute exacerbation of chronic obstructive pulmonary disease[J]. Chinese Journal of Practical Internal Medicine, 2009.
[14]贾宸.主旋律中提取若干关键问题的研究[D].北京:北京邮电大学,2018.
[15]廖俊杰.基于 Spark 的大数据人员信息智能管理系统[D].广东:广东工业大学,2018.
[16]闫晶.视频会议图像获取和处理技术研究[D].山西:中北大学,2008.
[17]谢静苑.基于特征图像分类以及稀疏表示的超分辨率重建[D].南京:南京邮电大学,2013.
[18]VUJOVIC V, MAKSIMOVIC M. Raspberry Pi as a wireless sensor node: performances and constraints[C]//International Convention on Information and Communication Technology, Electronics and Microelectronics.IEEE,2014:1013-1018.
[19]刘丽霞,边金松,穆森.基于 FFMPEG 解码的音视频同步实现[J].计算机工程与设计，2013,06:2087-2092.
[20]杨克伟.视频压缩算法研究[D].厦门:厦门大学,2014.
[21]雷霄骅,姜秀华,王彩虹.基于 RTMP 协议的流媒体技术的原理与应用[J].中国传媒大学学报(自然科学版),2013(6):59-64.
[22]肖鹏.H.264 压缩视频的超分辨率重建技术研究[D].南京:南京邮电大学,2011.
[23]老男孩.跟老男孩学 Linux 运维:Web 集群实战.北京:机械工业出版社,2016.

[24]王新蕾,刘乃丰,夏济海.基于 DirectShow 的视频处理 Filter 组件设计与实现[J].现代电子技术,2016,39(13):46-50.
[25]刘丹,王相海.视频帧差图像编码研究[J].计算机工程与应用,2007,07:45-48.
[26]刘斌.基于 FPGA 的 H.264 视频编码器[D].杭州:电子科技大学,2012.
[27]郭振.基于灰度投影的电子稳像系统及评价方法[D].天津:天津大学,2009.
[28]何书前,倪江群,石春.一种分层判决结构的 H.264/AVC 快速帧间模式选择方法[J].电子学报，2013,41(11):2199-2206.
[29]刘应涛.LTE 中广播多播服务的资源分配算法研究[D].成都:电子科技大学,2016.
[30]严新民.基于 DSP 的音视频编解码技术的研究[D].苏州:苏州大学,2005.
[31]吴炜,常义林.在接收端实现的媒体同步控制算法[J].系统工程与电子技术,2006,10:1587-1591.
[32]边朝政.用于光通信的视频编码器的研究[J].科技资讯,2013,33:12-13.
[33]朱剑英.基于 DCT 变换的图像编码方法研究[D].南京:南京理工大学,2004.
[34]王利萍.基于 Nginx 服务器集群负载均衡技术的研究与改进[D].山东:山东大学,2015.

攻读硕士学位期间发表学术论文情况

1 王洪玉，**席铮**. 无线投影方法及装置.专利申请号：201710305854.4

2 王洪玉，**席铮**. 通用多屏智能信息推送方法及装置.专利申请号：201710669367.6

3 王洪玉，**席铮**. 一种基于公共场所的智能广告信息实时推送方法及装置.专利申请号：201711432968.1

致　　谢

日月如磨蚁，万事且浮休。时间昼夜不曾停歇，此刻写到毕业设计致谢环节的我不由得愈加感慨，我即将与大连理工告别，即将与我的导师同学告别，即将与我这两年多的研究生时光告别。回眸沉思，我们都在不停的邂逅与告别中感叹着，成长着，收获着。一路走来，我拥有了太多人的帮助与关怀，导师、家人、好友、同窗，他们是我最珍视的收获，也是我此时此刻最想道一声感谢的人。

若要问我这两年研究生生活对我产生影响最大的人是谁，那回答必是我的导师——王洪玉教授。于我而言，他不仅是一名兢兢业业、认真做学术的老师，更为我树立了学习与生活中的榜样。王老师对待学术一丝不苟，当我们在做实验时或者解决问题时，他常常耐心地指导我们，为我们提供思路，教授我们方法。并且指导我获得了“歌尔杯”第三届全国研究生移动终端应用设计创新大赛特等奖。在这两年我若有一丝学术上的长进，都是承蒙恩师的不吝赐教。

其次我想感谢我的同门刘孝雷师兄、苗鑫师兄和陶奎印师弟，在教研室的时光充实而快乐，无论是学习新知识还是做实验，我们共同努力，互帮互助，团结协作，在学习和生活中我们都是最信任彼此的伙伴。希望大家都能不断进步，毕业后可以过上令人艳羡的人生。

这一切也都离不开学校的栽培，我要感谢大连理工大学，是学校为我提供了丰富的学习资源，为我提供了学习和成长的平台，也祝愿学校越来越好。

两年时光转瞬即逝，但这期间的成长与收获是将是我一生珍视的财富，感恩！

大连理工大学学位论文独创性声明

作者郑重声明：所呈交的学位论文，是本人在导师的指导下进行研究工作所取得的成果。尽我所知，除文中已经注明引用内容和致谢的地方外，本论文不包含其他个人或集体已经发表的研究成果，也不包含其他已申请学位或其他用途使用过的成果。与我一同工作的同志对本研究所做的贡献均已在论文中做了明确的说明并表示了谢意。

若有不实之处，本人愿意承担相关法律责任。

学位论文题目：基于ffmpeg和SDL的智能录屏及播放系统

作 者 签 名：[signature] 日期：2018 年 12 月 12 日

大连理工大学学位论文版权使用授权书

本人完全了解学校有关学位论文知识产权的规定，在校攻读学位期间论文工作的知识产权属于大连理工大学，允许论文被查阅和借阅。学校有权保留论文并向国家有关部门或机构送交论文的复印件和电子版，可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印、或扫描等复制手段保存和汇编本学位论文。

学位论文题目：基于FFmpeg和SDL的智能录屏及播放系统

作者签名：庞铮　　日期：2018 年 12 月 12 日

导师签名：王洪玉　　日期：2018 年 12 月 12 日